Beginning Linux

Neil Schneider, James (Jim) G. Sack, Chris K. Young, Dave Kaupp, Sang Woo Kang

Copyright 2005

Contents

This is being developed by members of the Kernel-Panic Linux User Group, and is designed to be a tutorial, for beginning Linux users. Not a compendium, but enough to allow the beginning user to begin to operate their system, and access the system resources. So you have installed Linux and now you want to know how to use it. That's what this book is about. Learning something as complex as Linux is not easy without actually sitting at the system. So why not just sit down right now, at your system, with your favorite beverage, and try some of the things you will be reading. Don't spill on the keyboard!

Basic Linux Commands and Usage

Getting into the system.

Because Linux is a multi-user multi-tasking operating system, similar to Unix you must log into the system. The login system provides a certain degree of security by preventing unauthorized people from accessing your files. It also provides access to the system. Your login must be unique. You can have multiple logins for different purposes. If for example, you were using you Linux system for accounting, you might want a separate username for doing payroll, so you might have a login called payroll.

Remember that Linux is case sensitive. So "JOE" is not the same as"joe". Loging in verifies who is logging on to the system, and sets up and "environment" for the user. We'll cover environments later. If you login incorrectly, the system will not tell you that you typed in a wrong username or password. It will only return "login failed".

When you first boot a Linux system, a banner is displayed, identifying the system. You may see something like the following information.

> Red Hat Linux release 9.0 (Shrike)
> Kernel 2.4.20 on an i686
> localhost login:

If you type in your username, you get the following:

> password:

If it is set up, the Message Of The Day (motd) will be displayed. You might also see a message from the system "you have mail in /var/spool/mail". How to retrieve mail from the system is not a part of this tutorial, so you'll have to read elsewhere, for that information. You type in your password correctly and you get the following:

```
localhost $
```

This is the "prompt". If you entered you username, or your password incorrectly, you will be greeted by:

```
Login incorrect
localhost login:
```

Don't panic, try again. Likely you either mis-typed either your login name, or your password. Try again. You're not logging in as root, are you?

Adding users

If you haven't already done so, you should create a username on your system. Once you are logged in as root you can do the following:

```
# adduser joe
# passwd joe
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
#
```

After this you can log off as root (Ctrl-d) and login as the new user you created.

Setting and changing a password.

From time to time, you may have reason to change your password. you use the passwd command to change you password. Here's what a typical session would look like.

```
joe@localhost $ passwd
changing password for joe
(current) Unix password:
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
```

Logging out

There are a couple of methods to log out of your Linux systems. You can either use the "exit" command or use the key combination <Ctrl>-d (Hold down the Ctrl and d key at the same time.) to log out.

Virtual Consoles

A useful feature of Linux is the virtual consoles. When you first log into a Linux system you are on the first "virtual" console of the system. On a standard Linux system you have eight virtual consoles. This is configurable, but we won't discuss that right now. Using the three key combination <Ctrl><Alt>F(n), where F(n) is one of the function keys between F1 and F8, you can move between these virtual consoles. If you are in X when you do this, F7 becomes your X session, and you will see the error output from the X server, in virtual console 1 (VC1) when you go back to it. Virtual consoles allow you to do multiple things at the same time, in console mode.

File System

On some single-user operating systems, placement of files and directories, is at the whim of the system operator, and the software developer. In Linux and other Unix-like operating systems, the filesystem design has been agreed upon, by consensus, over time. This is important in a system designed to be multi-user. Lack of an agreed-upon standard, for file placement, would make the sharing of resources difficult, if not impossible. So a consensus has built over time, and we see these agreed upon attributes on Linux systems.

Understanding some of the basic file concepts on a Linux system, is one of the keys to understanding how to make the best uses of it. One concept that is fundamental to understanding Linux is that to the system, everything is a file. The tape drive is a file, the keboard, mouse and monitor are files, the serial port is a file.

Files in Linux have three important features or attributes standard in, standard out and error. The concept is deceptively simple and some would even say elegant. By creating simple files that have only a few, predictable actions, and then allowing these files to be strung together, we can create a powerful, yet flexible system. By taking the standard out of one file, and directing it to the standard in of another file, we can build powerful and complex systems to accomplish our task. So the file is the primary building block of the system. All directories branch from the root of the file system represented by the single character "/". The standard common subdirectories are as follows.

/bin is the primary binary directory, bin is designated as a directory on the system where binary executable files are. Nearly all the commands shown below are contained in the /bin directory. Also you will find other similar binary directories for a similar purpose, in /usr/bin and /usr/local/bin. Generally, the deeper in the directory tree you go, the more unique the system becomes. Often users will have their own ~/bin directory, where they keep custom scripts and binaries for their personal use.

/boot is where the kernel, related files and files necessary to boot the system are kept. If you compile a new kernel, you will place the kernel in this directory.

/dev the device directory. This points to one of the features of Linux. To the system, everything is a file. So nearly every hardware device in you system will have a file in this directory associated with it. Some of these include fd0 for the first floppy, hda and sda for the first ide and scsi hard drives.

/etc is where most all system wide configuration files are. Files to configure nameservices, web servers, Login, filesystem, ftp-server, name-server, newserver, filesystem, X, configuration files are all kept in /etc as well as most of the system initialization files. If your looking for a file to configure a service on your system, the first place to look is /etc. There are other etc directories on some systems /usr/etc, /usr/local/etc and /usr/X11R6/lib/X11/etc which are directories for configurations of other parts of the system.

/lib is where shared libraries are. These are the libraries that are shared among applications running under Linux. These libraries are shared among running applications to reduce program overhead, by only loading those functions needed at run time. And by making one copy of the libraries available to multiple running programs, memory is conserved.

/home is where the users home directories are located. For example, the home directory for user joe, would be located in /home/joe. A user can abbreviate the reference to their home directory with the tilde "~". For example joe could abbreviate a file in his home directory by refering to it by ~/filename. If joe had created a bin directory in his home directory, he could reach it by the command cd ~/bin/.

/mnt is by convention and usage the directory where temporary filesystems would be mounted. /opt is a commonly the directory where commercial applications will be installed.

/proc is a virtualized file system. If you were able to look at the filesystem on a Linux system, when it wasn't running, you would find nothing in the directory. The files and directories under /proc are a model of what is going on within the running kernel. Under special circumstances files and instructions can be echoed to the /proc filesystem to make changes to the running kernel.

/root is the home directory of the super-user root. The root user's environment is set up by files in this directory, and files reserved for the use of the super-user are kept in this directory. In most ways this is the same as and other user's home directory.

/sbin is the system binary directory. It contains binaries or programs used to administer the system, for creating filesystems, starting system services and other administrative tasks. Binaries in /sbin are similar across different Linux systems. There is a similar directory

/usr/sbin where binary programs for administering services specific to the installation are. Another directory /usr/local/sbin will contain binaries for custom installed programs, that are not a part of the Linux distribution.

/tmp is a temporary directory. Files that are not essential to the operation of the system, temporary scratch files, and other fast changing files are created in /tmp

/usr is known as the users directory. system wide applications, libraries and documentation that are for the most part installed as part of the distribution, are in this directory and it's subdirectories. This is the users shared resource directory.

/var contains many files that change often. The subdirectories and files may remain somewhat constant, but the contents of those files is dynamic. Cache files, spool files, log files, pid files are all to be found in /var. The print and mail spools are both in subdirectories of /var.

Commands

What is a command?

Command n. : Statement presented by a human and accepted by a computer in such a manner as to make the human feel as if he is in control.

A command in Linux is a sequence of words (non-blank characters) separated by spaces (blanks). The entire command is terminated by the newline character (Enter or Return). The first word is the command, the remaining words are arguments to the command. Since spaces are used to separate arguments, the arguments themselves must not contain spaces. If for some reason you have to have an argument containing a space (ie; a filename or directory name containing a space) the argument should be surrounded by double quotes (“), or with single quotes or apostrophes (’). Do not use the grave accent (‘) for this purpose. Apostrophe and grave are not the same character. Make sure you understand which is which.

Arguments to Commands

In general, arguments to Linux system commands are one of three things. Filname the name of a file which the command is to act upon. Filenames can consist of almost any character. In practice though, most file names consist of only letters A-Z,a-z, numbers 0-9 and some other characters like underscore _ period . and minus sign -. Linux is case sensitive, so Linux and LINUX would be different filenames. There are special “meta-characters” or “wild card characters” for file name matching. You can use a shorthand notation for working on groups of files at a time.

Option this is a literal, often preceded by a minus sign (-). Long form arguments in Linux are preceded by two minus signs (–). For example rm -f

and rm –force are the same command arguments the former short form the latter long form. An option modifies the behavior of a command in some way, or gives details in exactly how the command is to operate. Some options are not preceded by anything.

Expression usually a character string, which is used as input to the command. Some are regular expressions using "metacharacter" strings which are used for string matching.

The general form for arguments following a command is:

command options expression filename(s)

but the exact order required by each command should be found by referring to the documentation for the specific command.

What follows are the man pages from Linux followed by some simple usage examples.

Man Pages

Historically UNIX systems have kept the User's Manual, Programmers Manual and Administrator's Manual on-line collectively refered to as manual pages, or man pages. Linux follows this tradition. There is a move to replace the man pages with Texinfo ("info info" will tell you more about Texinfo), and so many of the GNU man pages have a statement similar to the following.

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source.

Man pages describe user commands, system calls, library functions, configuration file formats, devices, subroutines, and kernel internals, on the system. It is important for you to familiarize yourself with the use of man, since it is the most readily available source of information on how to use Linux. There is GUI interface under X called xman.

The man pages come in for a lot of criticism for being cryptic and taking for granted a lot of UNIX background. Each page contains information about a single command or file. The bottom of the page often contains references to other files and related commands. Sometimes these references help clarify, often not.

Man pages are divided into sections, according to what they are used for. User commands are in section 1, UNIX system calls are in section 2, subroutines 3, devices 4, and so on. Sometimes an entry in two sections can have the same name and sometimes you will see a number following the name of a man page. This

number is a reference to the man page section the command is found. There are hard copy version available in bookstores. These are usually subdivided into sections correpsonding to the the same sections as the commands. For a complete description of the man command with all it's options, see the man subsection of the Commands section below.

pwd

NAME

pwd - print name of current/working directory

SYNOPSIS

pwd {–help,–version}

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source.

This manual page documents the GNU version of pwd. pwd prints the fully resolved name of the current directory. That is, all components of the printed name will be actual directory names – none will be symbolic links.

Note that most Unix shells provide a built-in pwd command with similar functionality so the unadorned, interactive pwd command will usually execute the built-in version and not this one.

OPTIONS

–**help** Print a usage message on standard output and exit successfully.

–**version** Print version information on standard output then exit successfully.

EXAMPLES

```
$ pwd
/home/root
$
```

mkdir

NAME

mkdir - make directories

SYNOPSIS

mkdir [-p] [-m mode] [–parents] [–mode=mode] [–help] [–version] dir...

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source. This manual page documents the GNU version of mkdir.

mkdir creates a directory with each given name. By default, the mode of created directories is 0777 minus the bits set in the umask.

OPTIONS

-m, –mode mode Set the mode of created directories to mode, which is symbolic as in chmod and uses the default mode as the point of departure.

-p, –parents Ensure that each given directory exists. Create any missing parent directories for each argument. Parent directories

default to the umask modified by ‘u+wx’. Do not consider an argument directory that already exists to be an error.

–**help** Print a usage message on standard output and exit successfully.

–**version** Print version information on standard output then exit successfully.

EXAMPLES

```
$ ls
$ mkdir bin
$ ls
bin
$
```

cat

NAME

cat concatenate files and print on the standard output

SYNOPSIS

cat [-benstuvAET] [–number] [–number-nonblank] [–squeeze-blank] [–show-nonprinting] [–show-ends] [–show-tabs] [–show-all] [–help] [–version] [file...]

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source. This manual page documents the GNU version of cat. cat writes the contents of each given file, or the standard input if none are given or when a file named '-' is given, to the standard output.

OPTIONS

-b, –number-nonblank Number all non-blank output lines, starting with 1.

-e Equivalent to -vE.

-n, –number Number all output lines, starting with 1.

-s, –squeeze-blank Replace multiple adjacent blank lines with a single blank line.

-t Equivalent to -vT.

-u Ignored; for Unix compatibility.

-v, –show-nonprinting Display control characters except for LFD and TAB using '^' notation and precede characters that have the high bit set with 'M-'.

-A, –show-all Equivalent to -vET.

-E, –show-ends Display a '$' after the end of each line.

-T, –show-tabs Display TAB characters as '^I'.

–**help** Print a usage message and exit with a status code indicating success.

–**version** Print version information on standard output then exit.

EXAMPLES

```
$ cat thisfile
This is the text in thisfile
$
```

cd

NAME

cd - change working directory

SYNOPSIS

cd “dirName”

DESCRIPTION

Change the current working directory to dirName, or to the home directory (as specified in the HOME environment variable) if dirName is not given. Returns an empty string.

KEYWORDS

working directory

EXAMPLES

```
$ cd /usr/local
$ pwd
/usr/local
$ cd
$ pwd
/home/myhome
```

chmod

NAME

chmod - changes the permissions of a file or directory

SYNOPSIS

chmod [-Rcfv] [–recursive] [–changes] [–silent] [–quiet] [–verbose] [–help] [–version] mode file...

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source. This manual page documents the GNU version of chmod. chmod changes the permissions of each given file according to mode, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new permissions. The format of a symbolic mode is ‘[ugoa...][[+-=][rwxXs- tugo...]...][,...]’. Multiple symbolic operations can be given, separated

by commas. A combination of the letters 'ugoa' controls which users' access to the file will be changed: the user who owns it (u), other users in the file's group (g), other users not in the file's group (o), or all users (a). If none of these are given, the effect is as if 'a' were given, but bits that are set in the umask are not affected.

The operator '+' causes the permissions selected to be added to the existing permissions of each file; '-' causes them to be removed; and '=' causes them to be the only permissions that the file has. The letters 'rwxXstugo' select the new permissions for the affected users: read (r), write (w), execute (or access for directories) (x),execute only if the file is a directory or already has execute permission for some user (X), set user or group ID on execution (s), save program text on swap device (t), the permissions that the user who owns the file currently has for it (u), the permissions that other users in the file's group have for it (g), and the permissions that other users not in the file's group have for it (o).

A numeric mode is from one to four octal digits (0-7), derived by adding up the bits with values 4, 2, and 1. Any omitted digits are assumed to be leading zeros. The first digit selects the set user ID (4) and set group ID (2) and save text image (1) attributes. The second digit selects permissions for the user who owns the file: read (4), write (2), and execute (1); the third selects permissions for other users in the file's group, with the same values; and the fourth for other users not in the file's group, with the same values. chmod never changes the permissions of symbolic links; the chmod system call cannot change their permissions. This is not a problem since the permissions of symbolic links are never used. However, for each symbolic link listed on the command line, chmod changes the permissions of the pointed-to file. In contrast, chmod ignores symbolic links encountered during recursive directory traversals.

OPTIONS

-c, –changes Verbosely describe only files whose permissions actually change.

-f, –silent, –quiet Do not print error messages about files whose permissions cannot be changed.

-v, –verbose Verbosely describe changed permissions.

-R, –recursive Recursively change permissions of directories and their contents.

–**help** Print a usage message on standard output and exit successfully.

–**version** Print version information on standard output then exit successfully.

EXAMPLES

```
$ ls -l
-rw-r–r– 1 root root 0 Mar 26 19:22 firstfile
```

-rw-r–r– 1 root root 0 Mar 26 19:22 fourfile
-rw-r–r– 1 root root 0 Mar 26 19:22 lastfile
-rw-r–r– 1 root root 0 Mar 26 19:22 onefile
-rw-r–r– 1 root root 0 Mar 26 19:22 thatfile
-rw-r–r– 1 root root 29 Mar 26 19:23 thisfile
-rw-r–r– 1 root root 0 Mar 26 19:22 thosefiles
-rw-r–r– 1 root root 0 Mar 26 19:22 threefile
-rw-r–r– 1 root root 0 Mar 26 19:22 twofile
$ chmod +x firstfile
-rwxr-xr-x 1 root root 0 Mar 26 19:22 firstfile*
$ chmod 444 fourfile
-r–r–r– 1 root root 0 Mar 26 19:22 fourfile
$ chmod 555 lastfile
-r-xr-xr-x 1 root root 0 Mar 26 19:22 lastfile*
$ chmod 666 onefile
-rw-rw-rw- 1 root root 0 Mar 26 19:22 onefile
$ chmod 777 thatfile
-rwxrwxrwx 1 root root 0 Mar 26 19:22 thatfile
$ chmod 400 thisfile
-r——— 1 root root 29 Mar 26 19:23 thisfile
$ chmod 500 thosefiles
-r-x—— 1 root root 0 Mar 26 19:22 thosefiles
$ chmod 600 threefile
-rw———- 1 root root 0 Mar 26 19:22 threefile
$ chmod 700 twofile
-rwx—— 1 root root 0 Mar 26 19:22 twofile*
$ chmod 1400 firstfile
-r———T 1 root root 0 Mar 26 19:22 firstfile
$ chmod 2400 fourfile
-r—-S— 1 root root 0 Mar 26 19:22 fourfile
$ chmod 3400 lastfile
-r—-S–T 1 root root 0 Mar 26 19:22 lastfile

chown

NAME

chown - change the user and group ownership of files

SYNOPSIS

chown [-Rcfv] [–recursive] [–changes] [–help] [–version] [–silent] [–quiet] [–verbose] [user][:.][group] file...

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source. This manual page documents the GNU version of chown. chown changes the user and/or group ownership of each given file, according to its first non-option argument, which is interpreted as follows. If only a user name (or numeric user ID) is given, that user is made the owner of each given file, and the files' group is not changed. If the user name is followed by a colon or dot and a group name (or numeric group ID), with no spaces between them, the group ownership of the files is changed as well. If a colon or dot but no group name follows the user name, that user is made the owner of the files and the group of the files is changed to that user's login group. If the colon or dot and group are given, but the user name is omitted, only the group of the files is changed; in this case, chown performs the same function as chgrp.

OPTIONS

-c, –changes Verbosely describe only files whose ownership actually changes.

-f, –silent, –quiet Do not print error messages about files whose ownership cannot be changed.

-v, –verbose Verbosely describe ownership changes.

-R, –recursive Recursively change ownership of directories and their contents.

–**help** Print a usage message on standard output and exit successfully.

–**version** Print version information on standard output then exit successfully.

EXAMPLES

```
$ ls -l
-rw-rw-rw- 1 root root 0 Mar 26 19:22 firstfile
-rw-rw-rw- 1 root root 0 Mar 26 19:22 fourfile
-rw-rw-rw- 1 root root 0 Mar 26 19:22 lastfile
-rw-rw-rw- 1 root root 0 Mar 26 19:22 onefile
-rw-rw-rw- 1 root root 0 Mar 26 19:22 thatfile
-rw-rw-rw- 1 root root 29 Mar 26 19:23 thisfile
-rw-rw-rw- 1 root root 0 Mar 26 19:22 thosefiles
-rw-rw-rw- 1 root root 0 Mar 26 19:22 threefile
-rw-rw-rw- 1 root root 0 Mar 26 19:22 twofile
```

cp

NAME

cp copy files

SYNOPSIS

cp [options] source dest cp [options] source... directory Options: [-abdfilprsuvxPR] [-S backup-suffix] [-V {numbered,existing,sim- ple}] [–backup] [–no-dereference] [–force] [–interactive] [–one-file-system] [–preserve] [–recursive] [–update] [–verbose] [–suffix=backup-suffix] [–version-control={numbered,exist- ing,simple}] [–archive] [–parents] [–link] [–symbolic-link] [–help] [–version]

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source. This manual page documents the GNU version of cp. If the last argument names an existing directory, cp copies each other given file into a file with the same name in that directory. Otherwise, if only two files are given, it copies the first onto the second. It is an error if the last argument is not a directory and more than two files are given. By default, it does not copy directories.

OPTIONS

-a, –archive Preserve as much as possible of the structure and attributes of the original files in the copy. The same as -dpR.

-b, –backup Make backups of files that are about to be overwritten or removed.

-d, –no-dereference Copy symbolic links as symbolic links rather than copying the files that they point to, and preserve hard link relationships between source files in the copies.

-f, –force Remove existing destination files.

-i, –interactive Prompt whether to overwrite existing regular destination files.

-l, –link Make hard links instead of copies of non-directories.

-P, –parents Form the name of each destination file by appending to the target directory a slash and the specified name of the source file. The last argument given to cp must be the name of an existing directory. For example, the command ‘cp –parents a/b/c existing_dir’ copies the file a/b/c to existing_dir/a/b/c, creating any missing intermediate direc- tories.

-p, –preserve Preserve the original files' owner, group, permissions, and timestamps.

-r Copy directories recursively, copying all non-directories as if they were regular files.

-s, –symbolic-link Make symbolic links instead of copies of non-directories. All source filenames must be absolute (starting with '/') unless the destination files are in the current directory. This option produces an error message on systems that do not support symbolic links.

-u, –update Do not copy a nondirectory that has an existing destination with the same or newer modification time.

-v, –verbose Print the name of each file before copying it.

-x, –one-file-system Skip subdirectories that are on different filesystems from the one that the copy started on.

-R, –recursive Copy directories recursively.

–**help** Print a usage message on standard output and exit successfully.

–**version** Print version information on standard output then exit successfully.

-S, –suffix backup-suffix The suffix used for making simple backup files can be set with the SIMPLE_BACKUP_SUFFIX environment variable, which can be overridden by this option. If neither of those is given, the default is '~', as it is in Emacs.

-V, –version-control {numbered,existing,simple} The type of backups made can be set with the VERSION_CONTROL environment variable, which can be overridden by this option. If VERSION_CONTROL is not set and this option is not given, the default backup type is 'existing'. The value of the VERSION_CONTROL environment variable and the argument to this option are like the GNU Emacs 'version-control' variable; they also recognize synonyms that are more descriptive. The valid values are (unique abbreviations are accepted):

- 't' or 'numbered' Always make numbered backups.
- 'nil' or 'existing' Make numbered backups of files that already have them, simple backups of the others.
- 'never' or 'simple' Always make simple backups.

EXAMPLES

```
$ cat thisfile
This is the text in thisfile
$ cat thatfile
$
$ cp thisfile thatfile
cp: overwrite 'thatfile'? y
$ cat thatfile
This is the text in thisfile
```

df

NAME

df - summarize free disk space

SYNOPSIS

df [-aikPv] [-t fstype] [-x fstype] [–all] [–inodes] [–type=fstype] [–exclude-type=fstype] [–kilobytes] [–portabil- ity] [–print-type] [–help] [–version] [filename...]

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source. This manual page documents the GNU version of df. df displays the amount of disk space available on the filesystem containing each file name argument. If no file name is given, the space available on all currently mounted filesystems is shown. Disk space is shown in 1K blocks by default, unless the environment variable POSIXLY_CORRECT is set, in which case 512-byte blocks are used. If an argument is the absolute file name of a disk device node containing a mounted filesystem, df shows the space available on that filesystem rather than on the filesystem containing the device node (which is always the root filesystem). This version of df cannot show the space available on unmounted filesystems, because on most kinds of systems doing so requires very nonportable intimate knowledge of filesystem structures.

OPTIONS

-a, –all Include in the listing filesystems that have 0 blocks, which are omitted by default. Such filesystems are typically special-purpose pseudo-filesystems, such as automounter entries. On some systems, filesystems

of type “ignore” or “auto” are also omitted by default and included in the listing by this option.

-i, –inodes List inode usage information instead of block usage. An inode (short for “index node”) is a special kind of disk block that contains information about a file, such as its owner, permissions, timestamps, and location on the disk.

-k, –kilobytes Print sizes in 1K blocks instead of 512-byte blocks. This overrides the environment variable POSIXLY_CORRECT. -P, –portability Use the POSIX output format. This is like the default format except that the information about each filesystem is always printed on exactly one line; a mount device is never put on a line by itself. This means that if the mount device name is more than 20 characters long (as for some net- work mounts), the columns are misaligned.

-T, –print-type Print a type string for each filesystem. Any such printed filesystem type name may be used as an argument to either of the –type= or –exclude-type= options.

-t, –type=fstype Limit the listing to filesystems of type fstype. Multiple filesystem types can be shown by giving multiple -t options. By default, all filesystem types are listed.

-x, –exclude-type=fstype Limit the listing to filesystems not of type-T, –print-type Print a type string for each filesystem. Any such printed filesystem type name may be used as an argument to either of the –type= or –exclude-type= options.

-t, –type=fstype Limit the listing to filesystems of type fstype. Multiple filesystem types can be shown by giving multiple -t options. By default, all filesystem types are listed.

-x, –exclude-type=fstype Limit the listing to filesystems not of type fstype. Multiple filesystem types can be shown by giving multiple

-t options. By default, all filesystem types are listed.

-x, –exclude-type=fstype Limit the listing to filesystems not of

type fstype. Multi- ple filesystem types can be eliminated by giving multiple -x options. By default, all filesystem types are listed.

-v Ignored; for compatibility with System V versions of df.

–**help** Print a usage message on standard output and exit successfully.

–**version** Print version information on standard output then exit successfully.

EXAMPLES

```
$ df
Filesystem 1024-blocks Used Available Capacity Mounted on
/dev/sda1 870677 694211 131486 84% /
/dev/sda5 2314835 1368909 826244 62%
/usr /dev/sda6 101075 27046 68810 28% /var
$ df -i
Filesystem Inodes IUsed IFree %IUsed Mounted on
/dev/sda1 225280 3334 221946 1% /
/dev/sda5 600064 60812 539252 10% /usr
/dev/sda6 26104 343 25761 1% /var
```

du

NAME

du - summarize disk usage

SYNOPSIS

du [-abcklsxDLS] [–all] [–total] [–count-links] [–summarize] [–bytes] [–kilobytes] [–one-file-system] [–separate-dirs] [–dereference] [–dereference-args] [–help] [–version] [file- name...]

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source. This manual page documents the GNU version of du. du displays the amount of disk space used by each argument and for each subdirectory of directory arguments. The space is measured in 1K blocks by default, unless the environment variable POSIXLY_CORRECT is set, in which case 512-byte blocks are used.

OPTIONS

-a, –all Display counts for all files, not just directories.

-b, –bytes Print sizes in bytes.

-c, –total Write a grand total of all of the arguments after all arguments have been processed. This can be used to find out the disk usage of a directory, with some files excluded.

-k, –kilobytes Print sizes in kilobytes. This overrides the environment variable POSIXLY_CORRECT.

-l, –count-links Count the size of all files, even if they have appeared already in another hard link.

-s, –summarize Display only a total for each argument.

-x, –one-file-system Skip directories that are on different filesystems from the one that the argument being processed is on.

-D, –dereference-args Dereference symbolic links that are command line arguments. Does not affect other symbolic links. This is helpful for finding out the disk usage of directories like /usr/tmp where they are symbolic links.

-L, –dereference Dereference symbolic links (show the disk space used by the file or directory that the link points to instead of the space used by the link).

-S, –separate-dirs Count the size of each directory separately, not including the sizes of subdirectories.

–**help** Print a usage message on standard output and exit successfully.

–**version** Print version information on standard output then exit successfully.

BUGS

On BSD systems, du reports sizes that are half the correct values for files that are NFS-mounted from HP-UX systems. On HP-UX systems, it reports sizes that are twice the correct values for files that are NFS-mounted from BSD systems. This is due to a flaw in HP-UX; it also affects the HP-UX du program.

EXAMPLES

```
$ du -a
1 ./thisfile 1
./thatfile 0
./thosefiles 0
./onefile 0
./twofile 0
./threefile 0
./fourfile 0
./firstfile 0
./lastfile 3 .
$ du -s
3 .
```

head

NAME

head - output the first part of files

SYNOPSIS

head [-c N[bkm]] [-n N] [-qv] [–bytes=N[bkm]] [–lines=N] [–quiet] [–silent] [–verbose] [–help] [–version] [file...] head [-Nbcklmqv] [file...]

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source.

This manual page documents the GNU version of head. head prints the first part (10 lines by default) of each given file; it reads from standard input if no files are given or when a filename of '-' is encountered. If more than one file is given, it prints a header consisting of the file's name enclosed in '==>' and '<==' before the output for each file.

OPTIONS

head accepts two option formats: the new one, in which numbers are arguments to the option letters, and the old one, in which the number precedes any option letters.

-c N, –bytes N Print first N bytes. N is a nonzero integer, optionally followed by one of the following characters to specify a different unit.

b 512-byte blocks.

k 1-kilobyte blocks.

m 1-megabyte blocks.

-n N, –lines N Print first N lines.

-q, –quiet, –silent Never print filename headers.

-v, –verbose Always print filename headers.

–**help** Print a usage message and exit with a status code indicating success.

–**version** Print version information on standard output then exit.

EXAMPLES

```
$ head /etc/services
#
# services This file describes the various services that are
# available from the TCP/IP subsystem. It should be
# consulted instead of using the numbers in the ARPA
# include files, or, worse, just guessing them.
#
# Version: @(#)/etc/services 2.00 04/30/93
#
# Author: Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
#
#
$
```

less

NAME

less - opposite of more

SYNOPSIS

less -? less –help less -V less –version less [-[+]aBcCdeEfgGiImMnNqQrsSuUVwX] [-b bufs] [-h lines] [-j line] [-k keyfile] [-{oO} logfile] [-p pattern] [-P prompt] [-t tag] [-T tagsfile] [-x tab] [-y lines] [-[z] lines] [+[+]cmd] [–] [filename]...

DESCRIPTION

Less is a program similar to more (1), but which allows backward movement in the file as well as forward movement. Also, less does not have to read the entire input file before starting, so with large input files it starts up faster than text editors like vi (1). Less uses termcap (or terminfo on some systems), so it can run on a variety of terminals. There is even limited support for hardcopy terminals. (On a hardcopy terminal, lines which should be printed at the top of the screen are prefixed with a caret.) Commands are based on both more and vi. Commands may be preceded by a decimal number, called N in the descriptions below. The number is used by some commands, as indicated.

COMMANDS

In the following descriptions, ^X means control-X. ESC stands for the ESCAPE key; for example ESC-v means the two character sequence "ESCAPE", then "v". h or H Help: display a summary of these commands. If you forget all the other commands, remember this one. SPACE or ^V or f or ^F Scroll forward N lines, default one window (see option -z below). If N is more than the screen size, only the final screenful is displayed. Warning: some systems use ^V as a special literalization character.

z Like SPACE, but if N is specified, it becomes the new window size.

ESC-SPACE Like SPACE, but scrolls a full screenful, even if it reaches end-of-file in the process.

RETURN or ^N or e or ^E or j or ^J Scroll forward N lines, default 1. The entire N lines are displayed, even if N is more than the screen size.

d or ^D Scroll forward N lines, default one half of the screen size. If N is specified, it becomes the new default for subsequent d and u commands.

b or ^B or ESC-v Scroll backward N lines, default one window (see option -z below). If N is more than the screen size, only the final screenful is displayed.

w Like ESC-v, but if N is specified, it becomes the new window size.

y or ^Y or ^P or k or ^K Scroll backward N lines, default 1. The entire N lines are displayed, even if N is more than the screen size. Warning: some systems use ^Y as a special job control character.

u or ^U Scroll backward N lines, default one half of the screen size. If N is specified, it becomes the new default for subsequent d and u commands.

ESC-> or RIGHTARROW Scroll horizontally right N characters, default 8. This behaves best if you also set the -S option (chop lines). Note that if you wish to enter a number N, you must use ESC-), not RIGHTARROW, because the arrow is taken to be a line editing command (see the LINE EDITING section).

ESC-< or LEFTARROW Scroll horizontally left N characters, default 8.

r or ^R or ^L Repaint the screen.

R Repaint the screen, discarding any buffered input. Useful if the file is changing while it is being viewed.

u or ^U Scroll backward N lines, default one half of the screen size. If N is specified, it becomes the new default for subsequent d and u commands.

F Scroll forward, and keep trying to read when the end of file is reached. Normally this command would be used when already at the end of the file. It is a way to monitor the tail of a file which is growing while it is being viewed. (The behavior is similar to the "tail -f" command.)

g or < or ESC-< Go to line N in the file, default 1 (beginning of file). (Warning: this may be slow if N is large.)

G or > or ESC-> Go to line N in the file, default the end of the file. (Warning: this may be slow if N is large, or if N is not specified and standard input, rather than a file, is being read.)

p or % Go to a position N percent into the file. N should be between 0 and 100. { If a left curly bracket appears in the top line displayed on the screen, the { command will go to the matching right curly bracket. The matching right curly bracket is positioned on the bottom line of the screen. If there is more than one left curly bracket on the top line, a number N may be used to specify the N-th bracket on the line. } If a right curly bracket appears in the bottom line displayed on the screen, the } command will go to the matching left curly bracket. The matching left curly bracket is positioned on the top line of the screen. If there is more than one right curly bracket on the top line, a number N may be used to specify the N-th bracket on the line.

(Like {, but applies to parentheses rather than curly brackets.

) Like }, but applies to parentheses rather than curly brackets.

[Like {, but applies to square brackets rather than curly brackets.

] Like }, but applies to square brackets rather than curly brackets.

ESC-^F Followed by two characters, acts like {, but uses the two characters as open and close brackets, respectively. For example, "ESC ^F < >" could be used to go forward to the > which matches the < in the top displayed line.

ESC-^B Followed by two characters, acts like }, but uses the two characters as open and close brackets, respectively. For example, "ESC ^B < >" could be used to go backward to the < which matches the > in the bottom displayed line.

m Followed by any lowercase letter, marks the current position with that letter.

' (Single quote.) Followed by any lowercase letter, returns to the position which was previously marked with that letter. Followed by another single quote, returns to the position at which the last "large" movement command was executed. Followed by a ^ or $, jumps to the beginning or end of the file respectively. Marks are preserved when a new file is examined, so the ' command can be used to switch between input files.

^X^X Same as single quote.

/pattern Search forward in the file for the N-th line containing the pattern. N defaults to 1. The pattern is a regular expression, as recognized by ed. The search starts at the second line displayed (but see the -a and -j options, which change this). Certain characters are special if entered at the beginning of the pattern; they modify the type of search rather than become part of the pattern:

^N or ! Search for lines which do NOT match the pattern.

^E or * Search multiple files. That is, if the search reaches the beginning of the current file without finding a match, the search continues in the previous file in the command line list.

^F or @ Begin the search at the first line of the FIRST file in the command line list, regardless of what is currently displayed on the screen or the settings of the -a or -j options.

^K Highlight any text which matches the pattern on the current screen, but don't move to the first match (KEEP current position).

^R Don't interpret regular expression metacharacters; that is, do a simple textual comparison.

?pattern Search backward in the file for the N-th line containing the pattern. The search starts at the line immediately before the top line displayed. Certain characters are special as in the / command:

^N or ! Search for lines which do NOT match the pattern.

^E or * Search multiple files. That is, if the search reaches the beginning of the current file without finding a match, the search continues in the previous file in the command line list.

^F or @ Begin the search at the last line of the last file in the command line list, regard- less of what is currently displayed on the screen or the settings of the -a or -j options.

^K As in forward searches.

^R As in forward searches.

ESC-/pattern Same as "/*".

ESC-?pattern Same as "?*". n Repeat previous search, for N-th line containing the last pattern. If the previous search was modified by ^N, the search is made for the N-th line NOT containing the pattern. If the previous search was modified by ^E, the search continues in the next (or previous) file if not satisfied in the current file. If the previous search was modified by ^R, the search is done without using regular expressions. There is no effect if the previous search was modified by ^F or ^K.

N Repeat previous search, but in the reverse direction.

ESC-n Repeat previous search, but crossing file boundaries. The effect is as if the previous search were modified by *.

ESC-N Repeat previous search, but in the reverse direction and crossing file boundaries.

ESC-u Undo search highlighting. Turn off highlighting of strings matching the current search pattern. If highlighting is already off because of a previous ESC-u command, turn highlighting back on. Any search command will also turn highlighting back on. (Highlighting can also be disabled by toggling the -G option; in that case search commands do not turn highlighting back on.)

:e [**filename**] Examine a new file. If the filename is missing, the "current" file (see the :n and :p commands below) from the list of files in the command line is re-examined. A percent sign (%) in the filename is replaced by the name of the current file. A pound sign (#) is replaced by the name of the previously examined file. However, two consecutive percent signs are simply replaced with a single percent sign. This allows you to enter a filename that contains a percent sign in the name. Similarly, two consecutive pound signs are replaced with a single pound sign. The filename is inserted into the command line list of files so that it can be seen by subsequent :n and :p commands. If the filename consists of several files, they are all inserted into the list of files and the first one is examined. If the filename contains one or more spaces, the entire filename should be enclosed in double quotes (also see the -" option).

^X^V or E Same as :e. Warning: some systems use ^V as a special literalization character. On such systems, you may not be able to use ^V.

:n Examine the next file (from the list of files given in the command line). If a number N is specified, the N-th next file is examined.

:p Examine the previous file in the command line list. If a number

N is specified, the N-th previous file is examined.

:x Examine the first file in the command line list. If a number N is specified, the N-th file in the list is examined.

= or ^G or :f Prints some information about the file being viewed, including its name and the line number and byte offset of the bottom line being displayed. If possible, it also prints the length of the file, the number of lines in the file and the percent of the file above the last displayed line.

- Followed by one of the command line option letters (see below), this will change the setting of that option and print a message describing the new setting. If the option letter has a numeric value (such as -b or -h), or a

string value (such as -P or -t), a new value may be entered after the option letter. If no new value is entered, a message describing the current setting is printed and nothing is changed.

-+ Followed by one of the command line option letters (see below), this will reset the option to its default setting and print a message describing the new setting. (The "-+X" command does the same thing as "-+X" on the command line.) This does not work for string-valued options.

– Followed by one of the command line option letters (see below), this will reset the option to the "opposite" of its default setting and print a message describing the new setting. (The "–X" command does the same thing as "-X" on the command line.) This does not work for numeric or string-valued options.

_ (Underscore.) Followed by one of the command line option letters (see below), this will print a message describing the current setting of that option. The setting of the option is not changed.

+cmd Causes the specified cmd to be executed each time a new file is examined. For example, +G causes less to initially display each file starting at the end rather than the beginning.

V Prints the version number of less being run.

q or Q or :q or :Q or ZZ Exits less.

The following four commands may or may not be valid, depending on your particular installation.

v Invokes an editor to edit the current file being viewed. The editor is taken from the environment variable VISUAL if defined, or EDITOR if VISUAL is not defined, or defaults to "vi" if neither VISUAL nor EDITOR is defined. See also the discussion of LESSEDIT under the section on PROMPTS below.

! shell-command Invokes a shell to run the shell-command given. A

percent sign (%) in the command is replaced by the name of the current file. A pound sign (#) is replaced by the name of the previously examined file. "!!" repeats the last shell command. "!" with no shell command simply invokes a shell. On Unix systems, the shell is taken from the environment variable SHELL, or defaults to "sh". On MS- DOS and OS/2 systems, the shell is the normal command processor.

| <m> shell-command <m> represents any mark letter. Pipes a section of the input file to the given shell command. The section of the file to be piped is between the first line on the current screen and the position marked by the letter. <m> may also be ^ or $ to indicate beginning or end of file respectively. If <m> is . or newline, the current screen is piped.

s filename Save the input to a file. This only works if the input is a pipe, not an ordinary file.

OPTIONS

Command line options are described below. Most options may be changed while less is running, via the "-" command. Options are also taken from the environment variable "LESS". For example, to avoid typing "less -options ..." each time less is invoked, you might tell csh:

setenv LESS "-options"

or if you use sh:

LESS="-options"; export LESS

The environment variable is parsed before the command line, so command line options override the LESS environment variable. If an option appears in the LESS variable, it can be reset to its default on the command line by beginning the command line option with "-+". For options like -P which take a following string, a dollar sign ($) may be used to signal the end of the string.

-? This option displays a summary of the commands accepted by less (the same as the h command). (Depending on how your shell interprets the question mark, it may be necessary to quote the question mark, thus: "-\?".)

–**help** Same as -?.

-a Causes searches to start after the last line displayed on the screen, thus skipping all lines displayed on the screen. By default, searches start at the second line on the screen (or after the last found line; see the -j option).

-bn Specifies the number of buffers less will use for each file. Buffers are 1K, and by default 10 buffers are used for each file (except if the file is a pipe; see the -B option). The number n specifies a different number of buffers to use.

-B By default, when data is read from a pipe, buffers are allocated automatically as needed. If a large amount of data is read from the pipe, this can cause a large amount of memory to be allocated. The -B option disables this automatic allocation of buffers for pipes, so that only the number of buffers specified by the -b option are used. Warning: use of -B can result in erroneous display, since only the most recently viewed part of the file is kept in memory; any earlier data is lost.

-c Causes full screen repaints to be painted from the top line down. By default, full screen repaints are done by scrolling from the bottom of the screen.

-C The -C option is like -c, but the screen is cleared before it is repainted.

-d The -d option suppresses the error message normally displayed if the terminal is dumb; that is, lacks some important capability, such as the ability to clear the screen or scroll backward. The

-d option does not otherwise change the behavior of less on a dumb terminal).

-e Causes less to automatically exit the second time it reaches end-of-file. By default, the only way to exit less is via the "q" command.

-E Causes less to automatically exit the first time it reaches end-of-file.

-f Forces non-regular files to be opened. (A non-regular file is a directory or a device special file.) Also suppresses the warning message when a binary file is opened. By default, less will refuse to open non-regular files.

-g Normally, less will highlight ALL strings which match the last search command. The -g option changes this behavior to highlight only the particular string which was found by the last search command. This can cause less to run somewhat faster than the default.

-G The -G option suppresses all highlighting of strings found by search commands.

-hn Specifies a maximum number of lines to scroll back- ward. If it is necessary to scroll backward more than n lines, the screen is repainted in a forward direction instead. (If the terminal does not have the ability to scroll backward, -h0 is implied.)

-i Causes searches to ignore case; that is, uppercase and lowercase are considered identical. This option is ignored if any uppercase letters appear in the search pattern; in other words, if a pattern contains uppercase letters, then that search does not ignore case.

-I Like -i, but searches ignore case even if the pattern contains uppercase letters.

-jn Specifies a line on the screen where the "target" line is to be positioned. A target line is the object of a text search, tag search, jump to a line number, jump to a file percentage, or jump to a marked position. The screen line is specified by a number: the top line on the screen is 1, the next is 2, and so on. The number may be negative to specify a line relative to the bottom of the screen: the bottom line on the screen is -1, the second to the bottom is -2, and so on. If the -j option is used, searches begin at the line immediately after the target line. For example, if "-j4" is used, the target line is the fourth line on the screen, so searches begin at the fifth line on the screen.

-kfilename Causes less to open and interpret the named file as a lesskey (1) file. Multiple -k options may be specified. If the LESSKEY environment variable is set, or if a lesskey file is found in a standard place (see KEY BINDINGS), it is also used as a lesskey file.

-m Causes less to prompt verbosely (like more), with the percent into the file. By default, less prompts with a colon.

-M Causes less to prompt even more verbosely than more.

-n Suppresses line numbers. The default (to use line numbers) may cause less to run more slowly in some cases, especially with a very large input file. Suppressing line numbers with the -n option will avoid this problem. Using line numbers means: the line number will be displayed in the verbose prompt and in the = command, and the v command will pass the current line number to the editor (see also the discussion of LESSEDIT in PROMPTS below).

-N Causes a line number to be displayed at the beginning of each line in the display.

-ofilename Causes less to copy its input to the named file as it is being viewed. This applies only when the input file is a pipe, not an ordinary file. If the file already exists, less will ask for confirmation before overwriting it.

-Ofilename The -O option is like -o, but it will overwrite an existing file without asking for confirmation. If no log file has been specified, the -o and -O options can be used from within less to specify a log file. Without a file name, they will simply report the name of the log file. The "s" command is equivalent to specifying -o from within less.

-ppattern The -p option on the command line is equivalent to specifying +/pattern; that is, it tells less to start at the first occurrence of pattern in the file.

-Pprompt Provides a way to tailor the three prompt styles to your own preference. This option would normally be put in the LESS environment variable, rather than being typed in with each less command. Such an option must either be the last option in the LESS variable, or be terminated by a dollar sign. -Ps followed by a string changes the default (short) prompt to that string.

-Pm changes the medium (-m) prompt.

-PM changes the long (-M) prompt.

-Ph changes the prompt for the help screen.

-P= changes the message printed by the = command. All prompt strings consist of a sequence of letters and special escape sequences. See the section on PROMPTS for more details.

-q Causes moderately ”quiet” operation: the terminal bell is not rung if an attempt is made to scroll past the end of the file or before the beginning of the file. If the terminal has a ”visual bell”, it is used instead. The bell will be rung on certain other errors, such as typing an invalid character. The default is to ring the terminal bell in all such cases.

-Q Causes totally ”quiet” operation: the terminal bell is never rung.

-r Causes ”raw” control characters to be displayed. The default is to display control characters using the caret notation; for example, a control-A (octal 001) is displayed as ”ˆA”. Warning: when the -r option is used, less cannot keep track of the actual appearance of the screen (since this depends on how the screen responds to each type of control character). Thus, various display problems may result, such as long lines being split in the wrong place.

-s Causes consecutive blank lines to be squeezed into a single blank line. This is useful when viewing nroff output.

-S Causes lines longer than the screen width to be chopped rather than folded. That is, the remainder of a long line is simply discarded. The default is to fold long lines; that is, display the remainder on the next line.

-ttag The -t option, followed immediately by a TAG, will edit the file containing that tag. For this to work, there must be a file called ”tags” in the current directory, which was previously built by the ctags (1) command. This option may also be specified from within less (using the - command) as a way of examining a new file. The command ”:t” is equivalent to specifying -t from within less.

-Ttagsfile Specifies a tags file to be used instead of ”tags”.

-u Causes backspaces and carriage returns to be treated as printable characters; that is, they are sent to the terminal when they appear in the input.

-U Causes backspaces, tabs and carriage returns to be treated as control characters; that is, they are handled as specified by the -r option.By default, if neither -u nor -U is given, backspaces which appear adjacent to an underscore character are treated specially: the underlined text is displayed using the terminal’s hardware underlining capability. Also, backspaces which appear between two identical characters are treated specially: the overstruck text is printed using the terminal’s hardware boldface capability. Other backspaces are deleted, along with the preceding character. Carriage returns immediately followed by a newline are deleted. Other carriage returns are handled as specified by the -r option. Text which is overstruck or underlined can be searched for if neither -u nor -U is in effect.

-V Displays the version number of less.

–**version** Same as -V.

-w Causes blank lines to be used to represent lines past the end of the file. By default, a tilde character (˜) is used.

-xn Sets tab stops every n positions. The default for n is 8.

-X Disables sending the termcap initialization and deinitialization strings to the terminal. This is sometimes desirable if the deinitialization string does something unnecessary, like clearing the screen.

-yn Specifies a maximum number of lines to scroll for- ward. If it is necessary to scroll forward more than n lines, the screen is repainted instead. The -c or -C option may be used to repaint from the top of the screen if desired. By default, any forward movement causes scrolling.

-[z]n Changes the default scrolling window size to n lines. The default is one screenful. The z and w commands can also be used to change the window size. The "z" may be omitted for compatibility with more. If the number n is negative, it indicates n lines less than the current screen size. For example, if the screen is 24 lines, -z-4 sets the scrolling window to 20 lines. If the screen is resized to 40 lines, the scrolling window automatically changes to 36 lines.

-" Changes the filename quoting character. This may be necessary if you are trying to name a file which contains both spaces and quote characters. Followed by a single character, this changes the quote character to that character. Filenames containing a space should then be surrounded by that character rather than by double quotes. Followed by two characters, changes the open quote to the first character, and the close quote to the second character. Filenames containing a space should then be preceded by the open quote character and followed by the close quote character. Note that even after the quote characters are changed, this option remains -" (a dash followed by a double quote).

– A command line argument of "–" marks the end of option arguments. Any arguments following this are interpreted as filenames. This can be useful when viewing a file whose name begins with a "-" or "+".

+ If a command line option begins with +, the remainder of that option is taken to be an initial command to less. For example, +G tells less to start at the end of the file rather than the beginning, and +/xyz tells it to start at the first occurrence of "xyz" in the file. As a special case, +<number> acts like +<number>g; that is, it starts the display at the specified line number (however, see the caveat under the "g" command above). If the option starts with ++, the initial command applies to every file being viewed, not just the first one. The + command described previously may also be used to set (or change) an initial command for every file.

LINE EDITING

When entering command line at the bottom of the screen (for example, a filename for the :e command, or the pattern for a search command), certain keys can be used to manipulate the command line. Most commands have an alternate form in [brackets] which can be used if a key does not exist on a particular keyboard. (The bracketed forms do not work in the MS-DOS version.) Any of these special keys may be entered literally by preceding it with the "literal" character, either ^V or ^A. A backslash itself may also be entered literally by entering two backslashes.

LEFTARROW [**ESC-h**] Move the cursor one space to the left.

RIGHTARROW [**ESC-l**] Move the cursor one space to the right.

^LEFTARROW [**ESC-b or ESC-LEFTARROW**] (That is, CONTROL and LEFTARROW simultaneously.) Move the cursor one word to the left.

^RIGHTARROW [**ESC-w ESC-RIGHTARROW**] (That is, CONTROL and RIGHTARRO simultaneously.) Move the cursor one word to the right.

HOME [**ESC-0**] Move the cursor to the beginning of the line.

END [**ESC-$**] Move the cursor to the end of the line.

BACKSPACE Delete the character to the left of the cursor, or cancel the command if the command line is empty.

DELETE or [**ESC-x**] Delete the character under the cursor.

^BACKSPACE [**ESC-BACKSPACE**] (That is, CONTROL and BACKSPACE simultaneously.Delete the word to the left of the cursor.

^DELETE [**ESC-X or ESC-DELETE**] (That is, CONTROL and DELETE simultaneously.) Delete the word under the cursor.

UPARROW [**ESC-k**] Retrieve the previous command line.

DOWNARROW [**ESC-j**] Retrieve the next command line.

TAB Complete the partial filename to the left of the cursor. If it matches more than one filename, the first match is entered into the command line. Repeated TABs will cycle thru the other matching filenames. If the completed filename is a directory, a "/" is appended to the filename. (On MS- DOS systems, a "\" is appended.) The environment variable LESSSEPARATOR can be used to specify a different character to append to a directory name.

BACKTAB [**ESC-TAB**] Like, TAB, but cycles in the reverse direction thru the matching filenames.

^L Complete the partial filename to the left of the cursor. If it matches more than one filename, all matches are entered into the command line (if they fit).

^U (Unix) or ESC (MS-DOS) Delete the entire command line, or cancel the command if the command line is empty. If you have changed your line-kill character in Unix to some- thing other than ^U, that character is used instead of ^U.

KEY BINDINGS

You may define your own less commands by using the program lesskey (1) to create a lesskey file. This file specifies a set of command keys and an action associated with each key. You may also use lesskey to change the line-editing keys (see LINE EDITING), and to set environment variables. If the environment variable LESSKEY is set, less uses that as the name of the lesskey file. Otherwise, less looks in a standard place for the lesskey file: On Unix systems, less looks for a lesskey file called "$HOME/.less". On MS-DOS systems, less looks for a lesskey file called "$HOME/_less", and if it is not found there, then looks for a lesskey file called "_less" in any directory specified in the PATH environment variable. On OS/2 systems, less looks for a lesskey file called "$HOME/less.ini", and if it is not found, then looks for a lesskey file called "less.ini" in any directory specified in the INIT environment variable, and if it not found there, then looks for a lesskey file called "less.ini" in any directory specified in the PATH environment variable. See the lesskey manual page for more details.

INPUT PREPROCESSOR

You may define an "input preprocessor" for less. Before less opens a file, it first gives your input preprocessor a chance to modify the way the contents of the file are displayed. An input preprocessor is simply an executable program (or shell script), which writes the contents of the file to a different file, called the replacement file. The contents of the replacement file are then displayed in place of the contents of the original file. However, it will appear to the user as if the original file is opened; that is, less will display the original filename as the name of the current file.

An input preprocessor receives one command line argument, the original filename, as entered by the user. It should create the replacement file, and when finished, print the name of the replacement file to its standard output. If the input preprocessor does not output a replacement filename, less uses the original file, as normal.

The input preprocessor is not called when viewing standard input. To set up an input preprocessor, set the LESSOPEN environment variable to a command line which will invoke your input preprocessor. This command line should include

one occurrence of the string "%s", which will be replaced by the filename when the input preprocessor command is invoked. When less closes a file opened in such a way, it will call another program, called the input postprocessor, which may perform any desired clean-up action (such as deleting the replacement file created by LESSOPEN). This program receives two command line arguments, the original filename as entered by the user, and the name of the replacement file.

To set up an input postprocessor, set the LESSCLOSE environment variable to a command line which will invoke your input postprocessor. It may include two occurrences of the string "%s"; the first is replaced with the original name of the file and the second with the name of the replacement file, which was output by LESSOPEN.

For example, on many Unix systems, these two scripts will allow you to keep files in compressed format, but still let less view them directly:

lessopen.sh:

```
#! /bin/sh
case "$1" in *.Z) uncompress -c $1 >/tmp/less.$$ 2>/dev/null
if [ -s /tmp/less.$$ ];
then echo /tmp/less.$$
else rm -f /tmp/less.$$ fi ;;
esac
```

lessclose.sh:

```
#! /bin/sh rm $2
```

To use these scripts, put them both where they can be executed and set LESSOPEN="lessopen.sh %s", and LESSCLOSE="lessclose.sh %s %s". More complex LESSOPEN and LESSCLOSE scripts may be written to accept other types of compressed files, and so on. It is also possible to set up an input preprocessor to pipe the file data directly to less, rather than putting the data into a replacement file. This avoids the need to decompress the entire file before starting to view it. An input preprocessor that works this way is called an input pipe.

An input pipe, instead of writing the name of a replacement file on its standard output, writes the entire contents of the replacement file on its standard output. If the input pipe does not write any characters on its standard output, then there is no replacement file and less uses the origi nal file, as normal.

To use an input pipe, make the first character in the LESSOPEN environment variable a vertical bar (|) to sig- nify that the input preprocessor is an input pipe.

For example, on many Unix systems, this script will work like the previous example scripts:

lesspipe.sh:

```
#! /bin/sh
case "$1" in *.Z) uncompress -c $1 2>/dev/null
;; esac
```

To use this script, put it where it can be executed and set LESSOPEN="|lesspipe.sh %s". When an input pipe is used, a LESSCLOSE postprocessor can be used, but it is usually not necessary since there is no replacement file to clean up. In this case, the replacement file name passed to the LESSCLOSE postprocessor is "-".

NATIONAL CHARACTER SETS

There are three types of characters in the input file:

- normal characters can be displayed directly to the screen.
- control characters should not be displayed directly, but are expected to be found in ordinary text files (such as backspace and tab).
- binary characters should not be displayed directly and are not expected to be found in text files.

A "character set" is simply a description of which characters are to be considered normal, control, and binary. The LESSCHARSET environment variable may be used to select a character set. Possible values for LESSCHARSET are:

- ascii The default character set. BS, TAB, NL, CR, and formfeed are control characters, all chars with values between 127 and 255 are binary, and all others are normal.
- latin1 Selects the ISO 8859/1 character set. latin-1 is the same as ASCII, except characters between 161 and 255 are treated as normal characters.
- dos Selects a character set appropriate for MS-DOS.
- koi8-r Selects a Russian character set.
- next Selects a character set appropriate for NeXT computers.

In special cases, it may be desired to tailor less to use a character set other than the ones definable by LESSCHARSET. In this case, the environment variable LESSCHARDEF can be used to define a character set. It should be set to a string where each character in the string represents one character in the character set. The character "." is used for a normal character, "c" for control, and "b" for binary.

A decimal number may be used for repetition. For example, "bccc4b." would mean character 0 is binary, 1, 2 and 3 are control, 4, 5, 6 and 7 are binary, and 8 is normal. All characters after the last are taken to be the same as the last, so characters 9 through 255 would be normal. (This is an example, and does not necessarily represent any real character set.)

This table shows the value of LESSCHARDEF which is equivalent to each of the possible values for LESSCHARSET:

ascii	8bcccbcc18b95.b
dos	8bcccbcc12bc5b95.b.
ebcdic	5bc6bcc7bcc41b.9b7.9b5.b..8b6.10b6.b9.7b 9.8b8.17b3.3b9.7b9.8b8.6b10.b.b.b.
iso8859	8bcccbcc18b95.33b.
koi8-r	8bcccbcc18b95.b128.
latin1	8bcccbcc18b95.33b.
next	8bcccbcc18b95.bb125.bb

If neither LESSCHARSET nor LESSCHARDEF is set, but your system supports the setlocale interface, less will use setlocale to determine the character set. setlocale is controlled by setting the LANG or LC_CTYPE environment variables.

Control and binary characters are displayed in standout (reverse video). Each such character is displayed in caret notation if pos- sible (e.g. ^A for control-A). Caret notation is used only if inverting the 0100 bit results in a normal printable character. Otherwise, the character is displayed as a hex number in angle brackets.

This format can be changed by setting the LESSBINFMT environment variable. LESSBINFMT may begin with a "*" and one character to select the display attribute: "*k" is blinking, "*d" is bold, "*u" is underlined, "*s" is standout, and "*n" is normal. If LESSBINFMT does not begin with a "*", normal attribute is assumed. The remainder of LESSBINFMT is a string which may include one printf-style escape sequence (a % followed by x, X, o, d, etc.). For example, if LESSBINFMT is "*u[%x]", binary characters are displayed in underlined hexadecimal surrounded by brackets. The default if no LESSBINFMT is specified is "*d<%X>".

PROMPTS The -P option allows you to tailor the prompt to your preference. The string given to the -P option replaces the specified prompt string. Certain characters in the string are interpreted specially. The prompt mechanism is rather complicated to provide flexibility, but the ordinary user need not understand the details of constructing personalized prompt strings.

A percent sign followed by a single character is expanded according to what the following character is:

- %bX Replaced by the byte offset into the current input file. The b is followed by a single character (shown as X above) which specifies the line whose byte offset is to be used. If the character is a "t", the byte offset of the top line in the display is used, an "m" means use the middle line, a "b" means use the bottom line, a "B" means use the line just after the bottom line, and a "j" means use the "target" line, as specified by the -j option.
- %B Replaced by the size of the current input file.
- %E Replaced by the name of the editor (from the VISUAL environment variable, or the EDITOR environment variable if VISUAL is not defined). See the discussion of the LESSEDIT feature below.
- %f Replaced by the name of the current input file.
- %i Replaced by the index of the current file in the list of input files.
- %lX Replaced by the line number of a line in the input file. The line to be used is determined by the X, as with the %b option.
- %L Replaced by the line number of the last line in the input file.
- %m Replaced by the total number of input files.
- %pX Replaced by the percent into the current input file. The line used is determined by the X as with the %b option.
- %s Same as %B.
- %t Causes any trailing spaces to be removed. Usually used at the end of the string, but may appear anywhere.
- %x Replaced by the name of the next input file in the list.

If any item is unknown (for example, the file size if input is a pipe), a question mark is printed instead. The format of the prompt string can be changed depending on certain conditions. A question mark followed by a single character acts like an "IF": depending on the following character, a condition is evaluated. If the condition is true, any characters following the question mark and condition character, up to a period, are included in the prompt. If the condition is false, such characters are not included. A colon appearing between the question mark and the period can be used to establish an "ELSE": any characters between the colon and the period are included in the string if and only if the IF condition is false.

Condition characters (which follow a question mark) may be:

- ?a True if any characters have been included in the prompt so far.
- ?bX True if the byte offset of the specified line is known.

- ?B True if the size of current input file is known.
- ?e True if at end-of-file.
- ?f True if there is an input filename (that is, if input is not a
- pipe).
- ?lX True if the line number of the specified line is known.
- ?L True if the line number of the last line in the file is known.
- ?m True if there is more than one input file.
- ?n True if this is the first prompt in a new input file.
- ?pX True if the percent into the current input file of the spec-
- ified line is known.
- ?s Same as "?B".
- ?x True if there is a next input file (that is, if the current input file is not the last one).

Any characters other than the special ones (question mark, colon, period, percent, and backslash) become literally part of the prompt. Any of the special characters may be included in the prompt literally by preceding it with a backslash.

Some examples:

- ?f%f:Standard input. This prompt prints the filename, if known; otherwise the string "Standard input".
- ?f%f .?ltLine %lt:?pt%pt\%:?btByte %bt:-... This prompt would print the filename, if known. The filename is followed by the line number, if known, otherwise the percent if known, otherwise the byte offset if known. Otherwise, a dash is printed. Notice how each question mark has a matching period, and how the % after the %pt is included literally by escaping it with a backslash.
- ?n?f%f .?m(file %i of %m) ..?e(END) ?x- Next\: %x..%t This prints the filename if this is the first prompt in a file, followed by the "file N of N" message if there is more than one input file. Then, if we are at end-of-file, the string "(END)" is printed followed by the name of the next file, if there is one. Finally, any trailing spaces are truncated. This is the default prompt. For reference, here are the defaults for the other two prompts (-m and -M respectively). Each is broken into two lines here for readability only.

- ?n?f%f .?m(file %i of %m) ..?e(END) ?x- Next\: %x.: ?pB%pB\%:byte %bB?s/%s...%t ?f%f .?n?m(file %i of %m) ..?ltline %lt?L/%L. :byte %bB?s/%s. . ?e(END) ?x- Next\: %x.:?pB%pB\%..%t And here is the default message produced by the = command:
- ?f%f .?m(file %i of %m) .?ltline %lt?L/%L. . byte %bB?s/%s. ?e(END) :?pB%pB\%..%t The prompt expansion features are also used for another purpose: if an environment variable LESSEDIT is defined, it is used as the command to be executed when the v command is invoked. The LESSEDIT string is expanded in the same way as the prompt strings.

The default value for LESSEDIT is:

%E ?lm+%lm. %f Note that this expands to the editor name, followed by a + and the line number, followed by the file name. If your editor does not accept the "+linenumber" syntax, or has other differences in invocation syntax, the LESSEDIT variable can be changed to modify this default.

SECURITY

When the environment variable LESSSECURE is set to 1, less runs in a "secure" mode. This means these features are disabled:

- ! the shell command
- | the pipe command
- :e the examine command.
- v the editing command
- s -o log files
- -k use of lesskey files
- -t use of tags files
- metacharacters in filenames, such as *
- filename completion (TAB, ^L)

Less can also be compiled to be permanently in "secure" mode.

ENVIRONMENT VARIABLES

Environment variables may be specified either in the system environment as usual, or in a lesskey (1) file.

COLUMNS Sets the number of columns on the screen. Takes precedence over the number of columns specified by the TERM variable.

EDITOR The name of the editor (used for the v command).

HOME Name of the user's home directory (used to find a lesskey file on Unix systems).

INIT Name of the user's init directory (used to find a lesskey file on OS/2 systems).

LANG Language for determining the character set.

LC_CTYPE Language for determining the character set.

LESS Options which are passed to less automatically.

LESSBINFMT Format for displaying non-printable, non-control characters.

LESSCHARDEF Defines a character set.

LESSCHARSET Selects a predefined character set.

LESSCLOSE Command line to invoke the (optional) input-postprocessor.

LESSECHO Name of the lessecho program (default "lessecho"). The lessecho program is needed to expand metacharacters, such as * and ?, in filenames on Unix systems.

LESSEDIT Editor prototype string (used for the v command). See discussion under PROMPTS.

LESSKEY Name of the default lesskey(1) file.

LESSMETACHARS List of characters which are considered "metacharacters" by the shell.

LESSMETAESCAPE Prefix which less will add before each metacharacter in a command sent to the shell. If LESSMETAESCAPE is an empty string, commands containing metacharacters will not be passed to the shell.

LESSOPEN Command line to invoke the (optional) input-preprocessor.

LESSSECURE Runs less in "secure" mode. See discussion under SECURITY.

LESSSEPARATORString to be appended to a directory name in filename com- pletion.

LINES Sets the number of lines on the screen. Takes precedence over the number of lines specified by the TERM variable.

PATH User's search path (used to find a lesskey file on MS-DOS and OS/2 systems).

SHELL The shell used to execute the ! command, as well as to expand filenames.

TERM The type of terminal on which less is being run.

VISUAL The name of the editor (used for the v command).

SEE ALSO

lesskey(1)

WARNINGS

The = command and prompts (unless changed by -P) report the line number of the line at the top of the screen, but the byte and per- cent of the line at the bottom of the screen.

If the :e command is used to name more than one file, and one of the named files has been viewed previously, the new files may be entered into the list in an unexpected order.

On certain older terminals (the so-called "magic cookie" terminals), search highlighting will cause an erroneous display. On such terminals, search highlighting is disabled by default to avoid possible problems.

In certain cases, when search highlighting is enabled and a search pattern begins with a ^, more text than the matching string may be highlighted.

On some systems, setlocale claims that ASCII characters 0 thru 31 are control characters rather than binary characters. This causes less to treat some binary files as ordinary, non-binary files. To workaround this problem, set the environment variable LESSCHARSET to "ascii" (or whatever character set is appropriate).

COPYRIGHT

Copyright (c) 1984,1985,1989,1994,1995,1996 Mark Nudelman Comments to: markn@fog.net

locate

NAME

locate - list files in databases that match a pattern

SYNOPSIS

locate [-d path] [–database=path] [–version] [–help] pattern...

DESCRIPTION

This manual page documents the GNU version of locate. For each given pattern, locate searches one or more databases of file names and displays the file names that contain the pattern. Patterns can contain shell-style metacharacters: '*', '?', and '[]'. The metacharacters do not treat '/' or '.' specially. Therefore, a pattern 'foo*bar' can match a file name that contains 'foo3/bar', and a pattern '*duck*' can match a file name that contains 'lake/.ducky'. Patterns that contain metacharacters should be quoted to protect them from expansion by the shell.

If a pattern is a plain string – it contains no metacharacters – locate displays all file names in the database that contain that string anywhere. If a pattern does contain metacharacters, locate only displays file names that match the pattern exactly. As a result, patterns that contain metacharacters should usually begin with a '*', and will most often end with one as well. The exceptions are patterns that are intended to explicitly match the beginning or end of a file name.

The file name databases contain lists of files that were on the system when the databases were last updated. The system administrator can choose the file name of the default database, the frequency with which the databases are updated, and the directories for which they contain entries; see updatedb(1L).

OPTIONS

-d path, –database=path Instead of searching the default file name

database, search the file name databases in path, which is a colon-separated list of database file names. You can also use the environment variable

LOCATE_PATH to set the list of database files to search. The option overrides the environment variable if both are used. The file name database format changed starting with GNU find and locate version 4.0 to allow machines with different byte orderings to share the databases. This version of locate can automatically recognize and read databases produced for older versions of GNU locate or Unix versions of locate or find.

–**help** Print a summary of the options to locate and exit.

–**version** Print the version number of locate and exit.

ENVIRONMENT

LOCATE_PATH

Colon-separated list of databases to search.

SEE ALSO

find(1L), locatedb(5L), updatedb(1L), xargs(1L) Finding Files (on-line in Info, or printed)

ls

NAME

ls, dir, vdir - list contents of directories

SYNOPSIS

ls [-abcdfgiklmnpqrstuxABCFGLNQRSUX1] [-w cols] [-T cols] [-I pattern] [–all] [–escape] [–directory] [–inode] [–kilobytes] [–numeric-uid-gid] [–no-group] [–hide- control-chars] [–reverse] [–size] [–width=cols] [–tab- size=cols] [–almost-all] [–ignore-backups] [–classify] [–file-type] [–full-time] [–ignore=pattern] [–dereference] [–literal] [–quote-name] [–recursive] [–sort={none,time,size,extension}] [–format={long,verbose, commas, across, vertical, single-column}] [–time={atime,access,use,ctime,status}] [–help] [–ver- sion] [–color[={yes,no,tty}]] [–colour[={yes,no,tty}]] [name...]

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source.

This manual page documents the GNU version of ls. dir and vdir are versions of ls with different default output formats. These programs list each given file or directory name. Directory contents are sorted alphabetically. For ls, files are by default listed in columns, sorted vertically, if the standard output is a terminal; otherwise they are listed one per line. For dir, files are by default listed in columns, sorted vertically. For vdir, files are by default listed in long format.

OPTIONS

-a, –all List all files in directories, including all files that start with '.'.

-b, –escape Quote non-graphic characters in file names using alphabetic and octal backslash sequences like those used in C.

-c, –time=ctime, –time=status Sort directory contents according to the files' status change time instead of the modification time. If the long listing format is being used, print the status change time instead of the modification time.

-d, –directory List directories like other files, rather than listing their contents.

-f Do not sort directory contents; list them in what- ever order they are stored on the disk. The same as enabling -a and -U and disabling -l, -s, and -t.

–**full-time** List times in full, rather than using the standard abbreviation heuristics.

-g Ignored; for Unix compatibility.

-i, –inode Print the index number of each file to the left of the file name.

-k, –kilobytes If file sizes are being listed, print them in kilobytes. This overrides the environment variable POSIXLY_CORRECT.

-l, –format=long, –format=verbose In addition to the name of each file, print the file type, permissions, number of hard links, owner name, group name, size in bytes, and timestamp (the modification time unless other times are selected). For files with a time that is more than 6 months old or more than 1 hour into the future, the times- tamp contains the year instead of the time of day.

-m, –format=commas List files horizontally, with as many as will fit on each line, separated by commas.

-n, –numeric-uid-gid List the numeric UID and GID instead of the names.

-p Append a character to each file name indicating the file type.

-q, –hide-control-chars Print question marks instead of non-graphic characters in file names.

-r, –reverse Sort directory contents in reverse order.

-s, –size Print the size of each file in 1K blocks to the left of the file name. If the environment variable POSIXLY_CORRECT is set, 512-byte blocks are used instead.

-t, –sort=time Sort directory contents by timestamp instead of alphabetically, with the newest files listed first.

-u, –time=atime, –time=access, –time=use Sort directory contents according to the files' last access time instead of the modification time. If the long listing format is being used, print the last access time instead of the modification time.

-x, –format=across, –format=horizontal List the files in columns, sorted horizontally.

-A, –almost-all List all files in directories, except for '.' and '..'.

-B, –ignore-backups Do not list files that end with '~', unless they are given on the command line.

-C, –format=vertical List files in columns, sorted vertically.

-F, –classify Append a character to each file name indicating the file type. For regular files that are executable, append a '*'. The file type indicators are '/' for directories, '@' for symbolic links, '|' for FIFOs, '=' for sockets, and nothing for regular files.

-G, –no-group Inhibit display of group information in a long format directory listing.

-L, –dereference List the files linked to by symbolic links instead of listing the contents of the links.

-N, –literal Do not quote file names.

-Q, –quote-name Enclose file names in double quotes and quote non-graphic characters as in C.

-R, –recursive List the contents of all directories recursively.

-S, –sort=size Sort directory contents by file size instead of alphabetically, with the largest files listed first.

-U, –sort=none Do not sort directory contents; list them in whatever order they are stored on the disk. This option is not called

-f because the Unix ls -f option also enables -a and disables -l,

-s, and -t. It seems useless and ugly to group those unrelated things together in one option. Since this option doesn't do that, it has a different name.

-X, –sort=extension Sort directory contents alphabetically by file extension (characters after the last '.'); files with no extension are sorted first.

-1, –format=single-column List one file per line.

-w, –width cols Assume the screen is cols columns wide. The default is taken from the terminal driver if possible; otherwise the environment variable COLUMNS is used if it is set; otherwise the default is 80.

-T, –tabsize cols Assume that each tabstop is cols columns wide. The default is 8.

-I, –ignore pattern Do not list files whose names match the shell pat- tern pattern unless they are given on the command line. As in the shell, an initial '.' in a file- name does not match a wildcard at the start of pattern.

–color, –colour, –color=yes, –colour=yes Colorize the names of files depending on the type of file. See DISPLAY COLORIZATION below.

–color=tty, –colour=tty Same as –color but only if standard output is a terminal. This is very useful for shell scripts and command aliases, especially if your favorite pager does not support color control codes.

–color=no, –colour=no Disables colorization. This is the default. Provided to override a previous color option.

–help Print a usage message on standard output and exit successfully.

–version Print version information on standard output then exit successfully.

DISPLAY COLORIZATION

When using the –color option, this version of ls will colorize the file names printed according to the name and type of file. By default, this colorization is by type only, and the codes used are ISO 6429 (ANSI) compliant.

You can override the default colors by defining the environment variable LS_COLORS (or LS_COLOURS). The format of this variable is reminiscent of the termcap(5) file format; a colon-separated list of expressions of the form "xx=string", where "xx" is a two-character variable name.

The variables with their associated defaults are:

no 0 Normal (non-filename) text

fi 0 Regular file

di 32 Directory

ln 36 Symbolic link

pi 31 Named pipe (FIFO)

so 33 Socket

bd 44;37 Block device

cd 44;37 Character device

ex 35 Executable file

mi (none) Missing file (defaults to fi)

or (none) Orphaned symbolic link (defaults to ln)

lc \e[Left code

rc m Right code

ec (none) End code (replaces lc+no+rc)

You only need to include the variables you want to change from the default. File names can also be colorized based on filename extension. This is specified in the LS_COLORS variable using the syntax "*ext=string". For example, using ISO 6429 codes, to color all C-language source files blue you would specify "*.c=34". This would color all files ending in .c in blue (34) color.

Control characters can be written either in C-style \-escaped notation, or in stty-like ^-notation. The C- style notation adds \e for Escape, _ for a normal space character, and \? for Delete. In addition, the \ escape character can be used to override the default interpretation of \, ^, : and =.

Each file will be written as <lc> <color code> <rc> <file- name> <ec>. If the <ec> code is undefined, the sequence <lc> <no> <rc> will be used instead. This is generally more convenient to use, but less general. The left, right and end codes are provided so you don't have to type common parts over and over again and to support weird terminals; you will generally not need to change them at all unless your terminal does not use ISO 6429 color sequences but a different system.

If your terminal does use ISO 6429 color codes, you can compose the type codes (i.e. all except the lc, rc, and ec codes) from numerical commands separated by semicolons. The most common commands are:

0 to restore default color

1 for brighter colors

4 for underlined text

5 for flashing text

30 for black foreground

31 for red foreground

32 for green foreground

33 for yellow (or brown) foreground

34 for blue foreground

35 for purple foreground

36 for cyan foreground

37 for white (or gray) foreground

40 for black background

41 for red background

42 for green background

43 for yellow (or brown) background

44 for blue background

45 for purple background

46 for cyan background

47 for white (or gray) background

Not all commands will work on all systems or display devices. A few terminal programs do not recognize the default end code properly. If all text gets colorized after you do a directory listing, try changing the no and fi codes from 0 to the numerical codes for your standard fore- and back- ground colors.

BUGS

On BSD systems, the -s option reports sizes that are half the correct values for files that are NFS-mounted from HP- UX systems. On HP-UX systems, it reports sizes that are twice the correct values for files that are NFS-mounted from BSD systems. This is due to a flaw in HP-UX; it also affects the HP-UX ls program.

If there was a single standard for the English language it would not be necessary to support redundant spellings.

EXAMPLES

man

NAME

man - format and display the on-line manual pages

manpath - determine user's search path for man pages

SYNOPSIS

man [-acdfhkKtwW] [-m system] [-p string] [-C config_file] [-M path] [-P pager] [-S section_list] [section] name ...

DESCRIPTION

man formats and displays the on-line manual pages. This version knows about the MANPATH and (MAN)PAGER environment variables, so you can have your own set(s) of personal man pages and choose whatever program you like to display the formatted pages. If section is specified, man only looks in that section of the manual. You may also specify the order to search the sections for entries and which preprocessors to run on the source files via command line options or environment variables. If name contains a / then it is first tried as a filename, so that you can do man ./foo.5 or even man /cd/foo/bar.1.gz.

OPTIONS

-C config_file Specify the man.conf file to use; the default is /etc/man.config. (See man.conf(5).)

-M path Specify the list of directories to search for man pages. If no such option is given, the environment variable MANPATH is used. If no such environment variable is found, the default list is found by consulting /etc/man.config. An empty substring of MANPATH denotes the default list.

-P pager Specify which pager to use. This option overrides the MANPAGER environment variable, which in turn overrides the PAGER variable. By default, man uses /usr/bin/less-is.

-S section_list List is a colon separated list of manual sections to search. This option overrides the MANSECT environment variable.

-a By default, man will exit after displaying the first manual page it finds. Using this option forces man to display all the manual pages that match name, not just the first.

-c Reformat the source man page, even when an up-to- date cat page exists. This can be meaningful if the cat page was formatted for a screen with a different number of columns, or if the preformatted page is corrupted.

-d Don't actually display the man pages, but do print gobs of debugging information.

-D Both display and print debugging info.

-f Equivalent to whatis.

-h Print a one-line help message and exit.

-k Equivalent to apropos.

-K Search for the specified string in *all* man pages. Warning: this is probably very slow! It helps to specify a section. (Just to give a rough idea, on my machine this takes about a minute per 500 man pages.)

-m system Specify an alternate set of man pages to search based on the system name given.

-p string Specify the sequence of preprocessors to run before nroff or troff. Not all installations will have a full set of preprocessors. Some of the preprocessors and the letters used to designate them are: eqn (e), grap (g), pic (p), tbl (t), vgrind (v), refer (r). This option overrides the MANROFFSEQ environment variable.

-t Use /usr/bin/groff -Tps -mandoc to format the man- ual page, passing the output to stdout. The output from /usr/bin/groff -Tps -mandoc may need to be passed through some filter or another before being printed.

-w or –path Don't actually display the man pages, but do print the location(s) of the files that would be formatted or displayed. If no argument is given: display (on stdout) the list of directories that is searched by man for man pages. If manpath is a link to man, then "manpath" is equivalent to "man –path".

-W Like -w, but print file names one per line, without additional information. This is useful in shell commands like man -aW man | xargs ls -l

CAT PAGES

Man will try to save the formatted man pages, in order to save formatting time the next time these pages are needed. Traditionally, formatted versions of pages in DIR/manX are saved in DIR/catX, but other mappings from man dir to cat dir can be specified in /etc/man.config. No cat pages are saved when the required cat directory does not exist. It is possible to make man suid to a user man. Then, if a cat directory has owner man and mode 0755 (only writable by man), and the cat files have owner man and mode 0644 or 0444 (only writable by man, or not writable at all), no ordinary user can change the cat pages or put other files in the cat directory. If man is not made suid, then a cat directory should have mode 0777 if all users should be able to leave cat pages there.

The option -c forces reformatting a page, even if a recent cat page exists.

ENVIRONMENT

MANPATH If MANPATH is set, its value is used as the path to search for manual pages.

MANROFFSEQ If MANROFFSEQ is set, its value is used to determine the set of preprocessors run before running nroff or troff. By default, pages are passed through the table preprocessor before nroff.

MANSECT If MANSECT is set, its value is used to determine which manual sections to search.

MANWIDTH If MANWIDTH is set, its value is used as the width manpages should be displayed. Otherwise the pages may be displayed over the whole width of your screen.

MANPAGER If MANPAGER is set, its value is used as the name of the program to use to display the man page. If not, then PAGER is used. If that has no value either, /usr/bin/less -is is used.

LANG If LANG is set, its value defines the name of the subdirectory where man first looks for man pages. Thus, the command 'LANG=dk man 1 foo' will cause man to look for the foo man page in .../dk/man1/foo.1, and if it cannot find such a file, then in .../man1/foo.1, where ... is a directory on the search path.

NLSPATH, LC_MESSAGES, LANG The environment variables NLSPATH and LC_MESSAGES (or LANG when the latter does not exist) play a role in locating the message catalog. (But the English messages are compiled in, and for English no catalog is required.) Note that programs like col(1) called by man also use e.g. LC_CTYPE.

PATH PATH is used in the construction of the default search path for man pages.

SYSTEM SYSTEM is used to get the default alternate system name (for use with the -m option).

SEE ALSO

apropos(1), whatis(1), less(1), groff(1).

BUGS

The -t option only works if a troff-like program is installed. If you see blinking \255 or <AD> instead of hyphens, put 'LESS- CHARSET=latin1' in your environment.

EXAMPLES

more

NAME

more - file perusal filter for crt viewing

SYNOPSIS

more [-dlfpcsu] [-num] [+/ pattern] [+ linenum] [file ...]

DESCRIPTION

More is a filter for paging through text one screenful at a time. This version is especially primitive. Users should realize that less(1) provides more(1) emulation and extensive enhancements.

OPTIONS

Command line options are described below. Options are also taken from the environment variable MORE (make sure to precede them with a dash ("-")) but command line options will override them.

-num This option specifies an integer which is the screen size (in lines).

-d more will prompt the user with the message "[Press space to continue,

'q' to quit.]" and will display "[Press 'h' for instructions.]" instead of ringing the bell when an illegal key is pressed.

-l more usually treats ^L (form feed) as a special character, and will pause after any line that contains a form feed. The -l option will prevent this behavior.

-f Causes more to count logical, rather than screen lines (i.e., long lines are not folded).

-p Do not scroll. Instead, clear the whole screen and then display the text.

-c Do not scroll. Instead, paint each screen from the top, clearing the remainder of each line as it is displayed.

-s Squeeze multiple blank lines into one.

-u Suppress underlining.

+/ The +/ option specifies a string that will be searched for before each file is displayed.

+num Start at line number num.

COMMANDS

Interactive commands for more are based on vi(1). Some commands may be preceded by a decimal number, called k in the descriptions below. In the following descriptions, ^X means control-X. h or ? Help: display a summary of these commands. If you forget all the other commands, remember this one.

SPACE Display next k lines of text. Defaults to current screen size.

z Display next k lines of text. Defaults to current screen size.

Argument becomes new default.

RETURN Display next k lines of text. Defaults to 1. Argument becomes new default.

d or ^D Scroll k lines. Default is current scroll size, initially 11.

Argument becomes new default.

q or Q or INTERRUPT Exit.

s Skip forward k lines of text. Defaults to 1.

f Skip forward k screenfuls of text. Defaults to 1.

b or ^B Skip backwards k screenfuls of text. Defaults to 1.

' Go to place where previous search started.

= Display current line number.

/pattern Search for kth occurrence of regular expression. Defaults to 1.

n Search for kth occurrence of last r.e. Defaults to 1.

!<cmd> or :!<cmd> Execute <cmd> in a subshell

v Start up /usr/bin/vi at current line

^L Redraw screen

:n Go to kth next file. Defaults to 1.

:p Go to kth previous file. Defaults to 1.

:f Display current file name and line number

. Repeat previous command

ENVIRONMENT

More utilizes the following environment variables, if they exist:

MORE This variable may be set with favored options to more.

SHELL Current shell in use (normally set by the shell at login time).

TERM Specifies terminal type, used by more to get the terminal characteristics necessary to manipulate the screen.

SEE ALSO

vi(1) less(1)

AUTHORS

Eric Shienbrood, UC Berkeley Modified by Geoff Peck, UCB to add underlining, single spacing Modified by John Foderaro, UCB to add -c and MORE environment variable

HISTORY

The more command appeared in 3.0BSD. This man page documents more version 5 .19 (Berkeley 6/29/88), which is currently in use in the Linux community. Documentation was produced using several other versions of the man page, and extensive inspection of the source code.

EXAMPLES

mount

NAME

mount - mount a file system

SYNOPSIS

mount [-hV]

mount -a [-fFnrsvw] [-t vfstype] [-O optlist]

mount [-fnrsvw] [-o options [,...]] device | dir

mount [-fnrsvw] [-t vfstype] [-o options] device dir

DESCRIPTION

All files accessible in a Unix system are arranged in one big tree, the file hierarchy, rooted at /. These files can be spread out over several devices. The mount command serves to attach the file system found on some device to the big file tree. Conversely, the umount(8) command will detach it again.

The standard form of the mount command, is mount -t type device dir This tells the kernel to attach the file system found on device (which is of type type) at the directory dir. The previous contents (if any) and owner and mode of dir become invisible, and as long as this file system remains mounted, the pathname dir refers to the root of the file system on device.

Three forms of invocation do not actually mount anything: mount -h prints a help message; mount -V prints a version string; and just mount [-t type] lists all mounted file systems (of type type) - see below.

The proc file system is not associated with a special device, and when mounting it, an arbitrary keyword, such as proc can be used instead of a device specification. (The customary choice none is less fortunate: the error message 'none busy' from umount can be confusing.)

Most devices are indicated by a file name (of a block special device), like /dev/sda1, but there are other possibilities. For example, in the case of an NFS mount, device may look like knuth.cwi.nl:/dir.

The file /etc/fstab (see fstab(5)), may contain lines describing what devices are usually mounted where, using which options. This file is used in three ways:

- The command mount -a [-t type] (usually given in a bootscript) causes all file systems mentioned in fstab (of the proper type) to be mounted as indicated, except for those whose line contains the noauto keyword. Adding the -F option will make mount fork, so that the filesystems are mounted simultaneously.
- When mounting a file system mentioned in fstab, it suffices to give only the device, or only the mount point.
- Normally, only the superuser can mount file systems. However, when fstab contains the user option on a line, then anybody can mount the corresponding system.

Thus, given a line

```
/dev/cdrom /cd iso9660 ro,user,noauto,unhide
```

any user can mount the iso9660 file system found on his CDROM using the command

mount /dev/cdrom

or

mount /cd

For more details, see fstab(5).

The programs mount and umount maintain a list of currently mounted file systems in the file /etc/mtab. If no arguments are given to mount, this list is printed. When the proc filesystem is mounted (say at /proc), the files /etc/mtab and /proc/mounts have very similar contents. The former has somewhat more information, such as the mount options used, but is not necessarily up-to-date (cf. the -n option below). It is possible to replace /etc/mtab by a symbolic link to /proc/mounts, but some information is lost that way, and in particular working with the loop device will be less convenient.

OPTIONS

The full set of options used by an invocation of mount is determined by first extracting the options for the file system from the fstab table, then applying any options specified by the -o argument, and finally applying a -r or -w option, when present.

Options available for the mount command:

-V Output version.

-h Print a help message.

-v Verbose mode.

-a Mount all filesystems (of the given types) mentioned in fstab.

-F (Used in conjunction with -a.) Fork off a new incarnation of mount for each device. This will do the mounts on different devices or different NFS servers in parallel. This has the advantage that it is faster; also NFS timeouts go in parallel. A disadvantage is that the mounts are done in undefined order. Thus, you cannot use this option if you want to mount both /usr and /usr/spool.

-f Causes everything to be done except for the actual system call; if it's not obvious, this "fakes" mounting the file system. This option is useful in conjunction with the -v flag to determine what the mount command is trying to do. It can also be used to add entries for devices that were mounted ear- lier with the -n option.

-n Mount without writing in /etc/mtab. This is necessary for example when /etc is on a read-only file system.

-s Tolerate sloppy mount options rather than failing. This will ignore mount options not supported by a filesystem type. Not all filesystems support this option. This option exists for support of the Linux autofs-based automounter.

-r Mount the file system read-only. A synonym is -o ro.

-w Mount the file system read/write. This is the default. A synonym is -o rw.

-t vfstype The argument following the -t is used to indicate the file system type. The file system types which are currently supported are listed in linux/fs/filesystems.c: minix, ext, ext2, xiafs, hpfs, msdos, umsdos, vfat, proc, nfs, iso9660, smbfs, ncpfs, affs, ufs, romfs, sysv, xenix, coher- ent.
Note that the last three are equivalent and that xenix and coherent will be removed at some point in the future – use sysv instead.
Since kernel version 2.1.21 the types ext and xiafs do not exist anymore.
The type iso9660 is the default.
If no -t option is given, or if the auto type is specified, the superblock is probed for the filesystem type (minix, ext, ext2, xiafs, iso9660, romfs are supported). If this probe fails and /proc/filesystems exists, then all of the filesystems listed there will be tried, except for those that are labeled "nodev" (e.g., proc and nfs).
Note that the auto type may be useful for user- mounted floppies.
Warning: the probing uses a heuristic (the presence of appropriate 'magic'), and could recognize the wrong filesystem type.
More than one type may be specified in a comma separated list. The list of file system types can be prefixed with no to specify the file system types on which no action should be taken. (This can be meaningful with the -a option.)
For example, the command:

    mount -a -t nomsdos,ext

mounts all file systems except those of type msdos and ext.

-o Options are specified with a -o flag followed by a comma separated string of options. Some of these options are only useful when they appear in the /etc/fstab file. The following options apply to any file system that is being mounted:

- async All I/O to the file system should be done asynchronously.
- atime Update inode access time for each access. This is the default

- auto Can be mounted with the -a option.
- defaults Use default options: rw, suid, dev, exec, auto, nouser, and async.
- dev Interpret character or block special devices on the file system.
- exec Permit execution of binaries.
- noatime Do not update inode access times on this file system (e.g, for faster access on the news spool to speed up news servers).
- noauto Can only be mounted explicitly (i.e., the -a option will not cause the file system to be mounted).
- nodev Do not interpret character or block special devices on the file system.
- noexec Do not allow execution of any binaries on the mounted filesystem. This option might be useful for a server that has file systems containing binaries for architectures other than its own.
- nosuid Do not allow set-user-identifier or set- group-identifier bits to take effect.
  nouser Forbid an ordinary (i.e., non-root) user to mount the file system. This is the default.
- remount Attempt to remount an already-mounted file system. This is commonly used to change the mount flags for a file system, especially to make a readonly file system writeable.
- ro Mount the file system read-only.
- rw Mount the file system read-write.
- suid Allow set-user-identifier or set-group-identifier bits to take effect.
- sync All I/O to the file system should be done synchronously.
- user Allow an ordinary user to mount the file system. This option implies the options noexec, nosuid, and nodev (unless overridden by subsequent options, as in the option line user,exec,dev,suid).

FILESYSTEM SPECIFIC MOUNT OPTIONS

The following options apply only to certain file systems. We sort them by file system. They all follow the -o flag.

Mount options for affs

uid=value and gid=value Set the owner and group of the root of the file system (default: uid=gid=0, but with option uid or gid without specified value, the uid and gid of the current process are taken).

setuid=value and setgid=value Set the owner and group of all files.

mode=value Set the mode of all files to value & 0777 disregarding the original permissions. Add search permission to directories that have read permission. The value is given in octal.

protect Do not allow any changes to the protection bits on the file system.

usemp Set uid and gid of the root of the file system to the uid and gid of the mount point upon the first sync or umount, and then clear this option. Strange...

verbose Print an informational message for each successful mount.

prefix=string Prefix used before volume name, when following a link.

volume=string Prefix (of length at most 30) used before '/' when following a symbolic link.

reserved=value (Default: 2.) Number of unused blocks at the start of the device.

root=value Give explicitly the location of the root block.

bs=value Give blocksize. Allowed values are 512, 1024, 2048, 4096.

grpquota / noquota / quota / usrquota These options are accepted but ignored.

Mount options for coherent

None.

Mount options for ext

None. Note that the 'ext' file system is obsolete. Don't use it. Since Linux version 2.1.21 extfs is no longer part of the kernel source.

Mount options for ext2

The 'ext2' file system is the standard Linux file system. Due to a kernel bug, it may be mounted with random mount options (fixed in Linux 2.0.4).

bsddf / minixdf Set the behavior for the statfs system call. The minixdf behaviour is to return in the f_blocks field the total number of blocks of the file system, while the bsddf behaviour (which is the default) is to subtract the overhead blocks used by the ext2 file system and not available for file storage. Thus

```
% mount /k -o minixdf; df /k; umount /k
Filesystem 1024-blocks Used Available Capacity Mounted on
/dev/sda6 2630655 86954 2412169 3
% /k
% mount /k -o bsddf; df /k; umount /k
Filesystem 1024-blocks Used Available Capacity Mounted on
/dev/sda6 2543714 13 2412169 0
% /k
```

(Note that this example shows that one can add command line options to the options given in /etc/fstab.)

check check=normal check=strict Set checking level. When at least one of these options is set (and check=normal is set by default) the inodes and blocks bitmaps are checked upon mount (which can take half a minute or so on a big disk). With strict checking, block deallocation checks that the block to free is in the data zone.

check=none / nocheck No checking is done.

debug Print debugging info upon each (re)mount.

errors=continue / errors=remount-ro / errors=panic Define the behavior when an error is encountered. (Either ignore errors and just mark the file system erroneous and continue, or remount the file system read-only, or panic and halt the system.) The default is set in the filesystem superblock, and can be changed using tune2fs(8).

grpid or bsdgroups / nogrpid or sysvgroups These options define what group id a newly created file gets. When grpid is set, it takes the group id of the directory in which it is created; other- wise (the default) it takes the fsgid of the cur- rent process, unless the directory has the setgid bit set, in which case it takes the gid from the parent directory, and also gets the setgid bit set if it is a directory itself.

resgid=n and resuid=n The ext2 file system reserves a certain percentage of the available space (by default 5%, see mke2fs(8) and tune2fs(8)). These options determine who can use the reserved blocks. (Roughly: whoever has the specified uid, or belongs to the specified group.)

sb=n Instead of block 1, use block n as superblock. This could be useful when the filesystem has been dam- aged. Usually, copies of the superblock are found every 8192 blocks: in block 1, 8193, 16385, ... (Thus, one gets hundreds or even thousands of copies of the superblock on a big filesystem. I do not know of options to mke2fs that would cause fewer copies to be written.)

grpquota / noquota / quota / usrquota These options are accepted but ignored.

Mount options for fat

(Note: fat is not a separate filesystem, but a common part of the msdos, umsdos and vfat filesystems.)

blocksize=512 / blocksize=1024 Set blocksize (default 512).

uid=value and gid=value Set the owner and group of all files. (Default: the uid and gid of the current process.)

umask=value Set the umask (the bitmask of the permissions that are not present). The default is the umask of the current process. The value is given in octal.

check=value Three different levels of pickyness can be chosen:

r[elaxed] Upper and lower case are accepted and equivalent, long name parts are truncated (e.g. verylongname.foobar becomes verylong.foo), leading and embedded spaces are accepted in each name part (name and extension).

n[ormal] Like "relaxed", but many special characters (*, ?, <, spaces, etc.) are rejected. This is the default.

s[trict] Like "normal", but names may not contain long parts and special characters that are sometimes used on Linux, but are not accepted by MS-DOS are rejected. (+, =, spaces, etc.)

conv=b[inary] conv=t[ext] / conv=a[uto] The fat file system can perform CRLF<–>NL (MS-DOS text format to UNIX text format) conversion in the kernel. The following conversion modes are available:

binary no translation is performed. This is the default.

text CRLF<–>NL translation is performed on all files. auto CRLF<–>NL translation is performed on all files that don't have a "well-known binary" extension. The list of known extensions can be found at the beginning of fs/fat/misc.c (as of 2.0, the list is: exe, com, bin, app, sys, drv, ovl, ovr, obj, lib, dll, pif, arc, zip, lha, lzh, zoo, tar, z, arj, tz, taz, tzp, tpz, gz, tgz, deb, gif, bmp, tif, gl, jpg, pcx, tfm, vf, gf, pk, pxl, dvi). Programs that do computed seeks won't like in-kernel text conversion. Several people have had their data ruined by this translation. Beware! For file systems mounted in binary mode, a conversion tool (fromdos/todos) is available.

debug Turn on the debug flag. A version string and a list of file system parameters will be printed (these data are also printed if the parameters appear to be inconsistent).

fat=12 / fat=16 Specify either a 12 bit fat or a 16 bit fat. This overrides the automatic FAT type detection routine. Use with caution!

quiet Turn on the quiet flag. Attempts to chown or chmod files do not return errors, although they fail. Use with caution!

sys_immutable, showexec, dots, nodots, dotsOK=[yes|no] Various misguided attempts to force Unix or DOS conventions onto a FAT file system.

Mount options for hpfs

uid=value and gid=value Set the owner and group of all files. (Default: the uid and gid of the current process.)

umask=value Set the umask (the bitmask of the permissions that are not present). The default is the umask of the current process. The value is given in octal.

case=lower / case=asis Convert all files names to lower case, or leave them. (Default: case=lower.)

conv=binary / conv=text / conv=auto For conv=text, delete some random CRs (in particular, all followed by NL) when reading a file. For conv=auto, choose more or less at random between conv=binary and conv=text. For conv=binary, just read what is in the file. This is the default.

nocheck Do not abort mounting when certain consistency checks fail.

Mount options for iso9660

Normal iso9660 filenames appear in a 8.3 format (i.e., DOS-like restrictions on filename length), and in addition all characters are in upper case. Also there is no field for file ownership, protection, number of links, provision for block/character devices, etc.

Rock Ridge is an extension to iso9660 that provides all of these unix like features. Basically there are extensions to each directory record that supply all of the additional information, and when Rock Ridge is in use, the filesystem is indistinguishable from a normal UNIX file system (except that it is read-only, of course).

norock Disable the use of Rock Ridge extensions, even if available. Cf. map.

check=r[elaxed] / check=s[trict] With check=relaxed, a filename is first converted to lower case before doing the lookup. This is probably only meaningful together with norock and map=normal. (Default: check=strict.)

uid=value and gid=value Give all files in the file system the indicated user or group id, possibly overriding the information found in the Rock Ridge extensions. (Default: uid=0,gid=0.)

map=n[ormal] / map=o[ff] For non-Rock Ridge volumes, normal name translation maps upper to lower case ASCII, drops a trailing ';1', and converts ';' to '.'. With map=off no name translation is done. See norock. (Default: map=normal.)

mode=value For non-Rock Ridge volumes, give all files the indicated mode. (Default: read permission for everybody.) Since Linux 2.1.37 one no longer needs to specify the mode in decimal. (Octal is indicated by a leading 0.)

unhide Also show hidden and associated files.

block=[512|1024|2048] Set the block size to the indicated value. (Default: block=1024.)

conv=a[uto] / conv=b[inary] / conv=m[text] / conv=t[ext] (Default: conv=binary.) Since Linux 1.3.54 this option has no effect anymore. (And non-binary settings used to be very dangerous, often leading to silent data corruption.)

cruft If the high byte of the file length contains other garbage, set this mount option to ignore the high order bits of the file length. This implies that a file cannot be larger than 16MB. The 'cruft' option is set automatically if the entire CDROM has a weird size (negative, or more than 800MB). It is also set when volume sequence numbers other than 0 or 1 are seen.

Mount options for minix

None.

Mount options for msdos

See mount options for fat. If the msdos file system detects an inconsistency, it reports an error and sets the file system read-only. The file system can be made write- able again by remounting it.

Mount options for ncp

Just like nfs, the ncp implementation expects a binary argument (a struct ncp_mount_data) to the mount system call. This argument is constructed by ncpmount(8) and the current version of mount (2.6h) does not know anything about ncp.

Mount options for nfs

Instead of a textual option string, parsed by the kernel, the nfs file system expects a binary argument of type struct nfs_mount_data. The program mount itself parses the following options of the form 'tag=value', and puts them in the structure mentioned:

rsize=n, wsize=n, timeo=n, retrans=n, acregmin=n, acregmax=n, acdirmin=n,

acdirmax=n, actimeo=n, retry=n, port=n, mountport=n, mounthost=name,

mountprog=n, mountvers=n, nfsprog=n, nfsvers=n, namlen=n. The option addr=n is accepted but ignored. Also the following Boolean options, possibly preceded by no are recognized: bg, fg, soft, hard, intr, posix, cto, ac, tcp, udp, lock. For details, see nfs(5). Especially useful options include
rsize=8192,wsize=8192
This will make your nfs connection much faster than with the default buffer size of 1024.

hard The program accessing a file on a NFS mounted file system will hang when the server crashes. The process cannot be interrupted or killed unless you also specify intr. When the NFS server is back online the program will continue undisturbed from where it was.
This is probably what you want.

soft This option allows the kernel to time out if the nfs server is not responding for some time. The time can be specified with timeo=time. This option might be useful if your nfs server sometimes doesn't respond or will be rebooted while some process tries to get a file from the server. Usually it just causes lots of trouble.

nolock Do not use locking. Do not start lockd.

Mount options for proc

uid=value and gid=value These options are recognized, but have no effect as far as I can see.

Mount options for romfs

None.

Mount options for smbfs

Just like nfs, the smb implementation expects a binary argument (a struct smb_mount_data) to the mount system call. This argument is constructed by smbmount(8) and the current version of mount (2.6c) does not know anything about smb.

Mount options for sysv

None.

Mount options for ufs

None.

Mount options for umsdos

See mount options for msdos. The dotsOK option is explicitly killed by umsdos.

Mount options for vfat

First of all, the mount options for fat are recognized. The dotsOK option is explicitly killed by vfat. Further- more, there are uni_xlate Translate unhandled Unicode characters to special escaped sequences. This lets you backup and restore filenames that are created with any Unicode characters. Without this option, a '?' is used when no translation is possible. The escape character is ':' because it is otherwise illegal on the vfat filesystem. The escape sequence that gets used, where u is the unicode character, is: ':', (u & 0x3f), ((u> >6) & 0x3f), (u> >12).

posix Allow two files with names that only differ in case.

nonumtail First try to make a short name without sequence number, before trying name~num.ext.

Mount options for xenix

None.

Mount options for xiafs

None. Although nothing is wrong with xiafs, it is not used much, and is not maintained. Probably one shouldn't use it. Since Linux version 2.1.21 xiafs is no longer part of the kernel source.

THE LOOP DEVICE

One further possible type is a mount via the loop device. For example, the command

mount /tmp/fdimage /mnt -t msdos -o loop=/dev/loop3,blocksize=1024

will set up the loop device /dev/loop3 to correspond to the file /tmp/fdimage, and then mount this device on /mnt. This type of mount knows about three options, namely loop, offset and encryption, that are really options to losetup(8). If no explicit loop device is mentioned (but just an option '-o loop' is given), then mount will try to find some unused loop device and use that.

EXAMPLES

Mount a DOS floppy disk to /mnt/floppy

```
$ mount -t msdos /dev/fd0 /mnt/floppy
Mount a cd to /mnt/cdrom
$ mount -t iso9660 /dev/cdrom /mnt/cdrom
```

mv

NAME

mv - rename files

SYNOPSIS

mv [options] source dest mv [options] source... directory Options: [-bfiuv] [-S backup-suffix] [-V {numbered,existing,sim- ple}] [–backup] [–force] [–interactive] [–update] [–verbose] [–suffix=backup-suffix] [–version-con- trol={numbered,existing,simple}] [–help] [–version]

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source.

This manual page documents the GNU version of mv. If the last argument names an existing directory, mv moves each other given file into a file with the same name in that directory. Otherwise, if only two files are given, it moves the first onto the second. It is an error if the last argument is not a directory and more than two files are given. It can move only regular files across filesystems. If a destination file is unwritable, the standard input is a tty, and the -f or –force option is not given, mv prompts the user for whether to overwrite the file. If the response does not begin with 'y' or 'Y', the file is skipped.

OPTIONS

-b, –**backup** Make backups of files that are about to be removed.

-f, –**force** Remove existing destination files and never prompt the user.

-i, –**interactive** Prompt whether to overwrite each destination file that already exists. If the response does not begin with 'y' or 'Y', the file is skipped.

-u, –**update** Do not move a nondirectory that has an existing destination with the same or newer modification time.

-v, –**verbose** Print the name of each file before moving it.

–**help** Print a usage message on standard output and exit successfully.

–**version** Print version information on standard output then exit successfully.

-S, –**suffix** backup-suffix The suffix used for making simple backup files can be set with the SIMPLE_BACKUP_SUFFIX environment variable, which can be overridden by this option. If neither of those is given, the default is '~', as it is in Emacs.

-V, –**version-control** {numbered,existing,simple} The type of backups made can be set with the VERSION_CONTROL environment variable, which can be overridden by this option. If VERSION_CONTROL is not set and this option is not given, the default backup type is 'existing'. The value of the VERSION_CONTROL environment variable and the argument to this

option are like the GNU Emacs 'version-control' variable; they also recognize synonyms that are more descriptive. The valid values are (unique abbreviations are accepted):

't' or 'numbered' Always make numbered backups.

'nil' or' existing' Make numbered backups of files that already have them, simple backups of the others.

'never' or 'simple' Always make simple backups.

EXAMPLES

Rename thisfile to thatfile

```
$ mv thisfile thatfile
```

Move a file from one directory to another. You only have to give the path to the new place, the file name will be the same, unless you choose to change it.

```
$ mv thisfile /home/joe/
```

ps

NAME

ps - report process status

SYNOPSIS

ps [-] [lujsvmaxScewhrnu] [txx] [O[+|-]k1[[+|-]k2...]] [pids] there are also three long options:

–sortX[+|-]key[,[+|-]key[,...]]

–help

–version

More long options are on the way...

DESCRIPTION

ps gives a snapshot of the current processes. If you want a repetitive update of this status, use top. This man page documents the /proc-based version of ps, or tries to.

COMMAND-LINE OPTIONS

The command-line options for this version of ps are derived from the BSD version of ps, not the System V version. The command-line arguments should not be preceded by a '-' character, because in the future, a '-' will be used to indicate Unix98-standard command-line arguments, while no '-' will indicate the current "extended BSD" style of command line arguments.

For now, ps will give you a warning if you use a '-' for a short option, but it will still work. If you have shell scripts which use BSD-style arguments to ps, take heed of the warning and fix them, or else your scripts will fail to function correctly at some point in the future.

If you want to turn off the warnings, set the I_WANT_A_BROKEN_PS environment variable.

There are also some "long options" in GNU style; see below for those.

l long format

u user format: gives user name and start time

j jobs format: pgid sid

s signal format

v vm format

m displays memory info (combine with p flag to get number of pages).

f "forest" family tree format for command line a show processes of other users too

x show processes without controlling terminal

S add child cpu time and page faults

c command name from task_struct

e show environment after command line and ' + '

w wide output: don't truncate command lines to fit on one line. To be exact, every w that is specified will add another possible line to the output. If the space isn't needed it isn't used. You may up to 100 w's. h no header r running procs only n numeric output for USER and WCHAN.

txx only procs with controlling tty xx; for xx you may use either the name of a device file under "/dev" or that name with either tty or cu sliced off. This is the reverse heuristic that ps uses to print out the abbreviated tty name in the TT field, e.g. ps -t1.

O[+|-]k1[,[+|-]k2[,...]] Order the process listing according to the multilevel sort specified by the sequence of short keys from SORT KEYS, k1, k2, ... Default order specifications exist for each of the various formats of ps. These are over-ridden by a user specified ordering. The '+' is quite optional, merely re-iterating the default direction on a key. '-' reverses direction only on the key it precedes.
As with t and pids, the O option must be the last option in a single command argument, but specifications in successive arguments are catenated.

pids List only the specified processes; they are comma- delimited. The list must be given immediately after the last option in a single command-line argument, with no intervening space, e.g. ps -j1,4,5. Lists specified in subsequent arguments are catenated, e.g. ps -l 1,2 3,4 5 6 will list all of the processes 1-6 in long format. If pids are given, they are listed no matter what. If a tty is given matching processes are listed no matter what. These two features override the 'a' and 'x' flags.

LONG COMMAND-LINE OPTIONS

These options are preceded by a double-hyphen.

–sortX[+|-]key[,[+|-]key[,...]] Choose a multi-letter key from the SORT KEYS section. X may be any convenient separator character. To be GNU-ish use '='. The '+' is really optional since default direction is increasing numerical or lexicographic order. E.g.:

```
ps -jax –sort=uid,-ppid,+pid
```

–help Get a help message that summarizes the usage and gives a list of supported sort keys. This list may be more up to date than this man page.

–version Display version and source of this program.

SORT KEYS

Note that the values used in sorting are the internal values ps uses and not the 'cooked' values used in some of the output format fields.

SHORT LONG DESCRIPTION

c cmd simple name of executable

C cmdline full command line

f flags flags as in long format F field

g pgrp process group ID

G tpgid controlling tty process group ID

j cutime cumulative user time

J cstime cumulative system time

k utime user time

K stime system time

m min_flt number of minor page faults

M maj_flt number of major page faults

n cmin_flt cumulative minor page faults

N cmaj_flt cumulative major page faults

o session session ID

p pid process ID

P ppid parent process ID

r rss resident set size

R resident resident pages

s size memory size in kilobytes

S share amount of shared pages

t tty the minor device number of tty

T start_time time process was started

U uid user ID number

u user user name

v vsize total VM size in bytes

y priority kernel scheduling priority

FIELD DESCRIPTIONS

PRI This is the counter field in the task struct. It is the time in HZ of the process's possible timeslice.

NI Standard unix nice value; a positive value means less cpu time.

SIZE Virtual image size; size of text+data+stack.

RSS Resident set size; kilobytes of program in memory.

WCHAN Name of the kernel function where the process is sleeping, with the 'sys_' stripped from the function name. If /etc/psdatabase does not exist, it is just a hex number instead.

STAT Information about the status of the process. The first field is R for runnable, S for sleeping, D for uninterruptible sleep, T for stopped or traced, or Z for a zombie process. The second field contains W if the process has no resident pages. The third field is N if the process has a positive nice value (NI field).

TT Controlling tty.

PAGEIN Number of major page faults (page faults that cause pages to be read from disk, including pages read from the buffer cache).

TRS Text resident size.

SWAP Kilobytes (or pages if -p is used) on swap device.

SHARE Shared memory.

UPDATING This proc-based ps works by reading the files in the proc filesystem, mounted on /proc. This ps does not need to be suid kmem or have any privileges to run. Do not give this ps any special permissions.
You will need to update the /etc/psdatabase file by running /usr/sbin/psupdateto get meaningful information from the WCHAN field. This should be done every time you compile a new kernel. You should also run 'ps' as root once and then any time the tty devices in the "/dev" directory change.

As of procps-1.00, ps/top read System.map directly if it is available. The search path for kernel address-to-symbol resolution is:

- $PS_SYSTEM_MAP
- /boot/System.map-‘uname -r‘
- /boot/System.map
- /lib/modules/‘uname

- -r‘/System.map
- /etc/psdatabase
- /boot/psdatabase-‘uname -r‘
- /boot/psdatabase,
- /lib/modules/‘uname -r‘/psdatabase

NOTES

The member used_math of task_struct is not shown, since crt0.s checks to see if math is present. This causes the math flag to be set for all processes, and so it is worthless.

Programs swapped out to disk will be shown without command line arguments, and unless the c option is given, in parentheses.

%CPU shows the cputime/realtime percentage. It will not add up to 100% unless you are lucky. It is time used divided by the time the process has been running.

The SIZE and RSS fields don’t count the page tables and the task_struct of a proc; this is at least 12k of memory that is always resident. SIZE is the virtual size of the proc (code+data+stack).

To perform the device number to name mapping ps maintains a file called ”/etc/psdevtab” (updated whenever ”/dev” becomes newer and permissions allow update). If permissions do not allow update, every invokation of ps requires a stat(2) of every file in the ”/dev” directory. If ”/dev” entries change often on your system, you should run ps as root often. I may add a fallback file under $HOME under force of popular demand.

AUTHOR

ps was originally written by Branko Lankester <lankeste@fwi.uva.nl>. Michael K. Johnson <johnsonm@red- hat.com> re-wrote it significantly to use the proc filesystem, changing a few things in the process. Michael Shields <mjshield@nyx.cs.du.edu> added the pid-list feature. Charles Blake <cblake@bbn.com> added multi-level sorting, the dirent-style library, the device name-to-num- ber mmaped database, the approximate binary search directly on System.map, and many code and documentation cleanups. David Mossberger-Tang wrote the generic BFD support for psupdate. Michael K. Johnson <johnsonm@red- hat.com> is the current maintainer.

Please send bug reports to <procps-bugs@redhat.com>

EXAMPLES

```
joe@localhost $ ps
PID TTY STAT TIME COMMAND
3485 1 S 0:00 xterm -sb
3509 p1 S 0:00 -bash
16910 2 S 0:00 /sbin/mingetty tty2
17024 p4 S 0:00 -bash
17076 3 S 0:00 (mingetty)
17077 4 S 0:00 (mingetty)
17078 5 S 0:00 (mingetty)
17079 6 S 0:00 (mingetty)
17647 p4 R 0:00 ps
```

rm

NAME

rm - remove files

SYNOPSIS

rm [-dfirvR] [–directory] [–force] [–interactive] [–recursive]

[–help] [–version] [–verbose] name...

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source.

This manual page documents the GNU version of rm. rm removes each specified file. By default, it does not remove directories. If a file is unwritable, the standard input is a tty, and the -f or –force option is not given, rm prompts the user for whether to remove the file. If the response does not begin with ‘y’ or ‘Y’, the file is skipped.

GNU rm, like every program that uses the getopt function to parse its arguments, lets you use the – option to indicate that all following arguments are non-options. To remove a file called ‘-f’ in the current directory, you could type either

```
rm – -f
```

or

rm ./-f

The Unix rm program's use of a single '-' for this purpose predates the development of the getopt standard syntax.

OPTIONS

-d, –directory Remove directories with 'unlink' instead of 'rmdir', and don't require a directory to be empty before trying to unlink it. Only works for the super-user. Because unlinking a directory causes any files in the deleted directory to become unreferenced, it is wise to fsck the filesystem after doing this.

-f, –force Ignore nonexistent files and never prompt the user.

-i, –interactive Prompt whether to remove each file. If the response does not begin with 'y' or 'Y', the file is skipped.

-r, -R, –recursive Remove the contents of directories recursively.

-v, –verbose Print the name of each file before removing it.

–help Print a usage message on standard output and exit successfully.

–version Print version information on standard output then exit successfully

EXAMPLES

```
joe@localhost $ ls -l
-rw-rw-rw- 1 joe joe 0 Mar 26 19:22 firstfile
-rw-rw-rw- 1 joe joe 0 Mar 26 19:22 fourfile
-rw-rw-rw- 1 joe joe 0 Mar 26 19:22 lastfile
-rw-rw-rw- 1 joe joe 0 Mar 26 19:22 onefile
joe@localhost $ rm onefile
rm: remove 'onefile'? y
joe@localhost $ ls -l
-rw-rw-rw- 1 joe joe 0 Mar 26 19:22 firstfile
-rw-rw-rw- 1 joe joe 0 Mar 26 19:22 fourfile
-rw-rw-rw- 1 joe joe 0 Mar 26 19:22 lastfile
```

rmdir

NAME

rmdir - remove empty directories

SYNOPSIS

rmdir [-p] [–parents] [–help] [–version] dir...

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source.

This manual page documents the GNU version of rmdir. rmdir removes each given empty directory. If any non-option argument does not refer to an existing empty directory, it is an error.

OPTIONS

-p, –**parents** Remove any parent directories that are explicitly mentioned in an argument, if they become empty after the argument file is removed.

–**help** Print a usage message on standard output and exit successfully.

–**version** Print version information on standard output then exit successfully.

EXAMPLES

shutdown

NAME

shutdown - bring the system down

SYNOPSIS

/sbin/shutdown [-t sec] [-rkhncfF] time [warning-message]

DESCRIPTION

shutdown brings the system down in a secure way. All logged-in users are notified that the system is going down, and login(1) is blocked. It is possible to shut the system down immediately or after a specified delay. All processes are first notified that the system is going down by the signal SIGTERM. This gives programs like vi(1) the time to save the file being edited, mail and news processing programs a chance to exit cleanly, etc. shutdown does its job by signalling the init process, asking it to change the runlevel. Runlevel 0 is used to halt the system, runlevel 6 is used to reboot the system, and runlevel 1 is

used to put to system into a state where administrative tasks can be performed; this is the default if neither the -h or -r flag is given to shutdown. To see which actions are taken on halt or reboot see the appropriate entries for these runlevels in the file /etc/inittab.

OPTIONS

-t sec Tell init(8) to wait sec seconds between sending processes the warning and the kill signal, before changing to another runlevel.

-k Don't really shutdown; only send the warning messages to everybody.

-r Reboot after shutdown.

-h Halt after shutdown.

-n [DEPRECATED] Don't call init(8) to do the shutdown but do it ourself. The use of this option is discouraged, and its results are not always what you'd expect.

-f Skip fsck on reboot.

-F Force fsck on reboot.

-c Cancel an already running shutdown. With this option it is of course not possible to give the time argument, but you can enter a explanatory message on the command line that will be sent to all users.

time When to shutdown.

warning-message Message to send to all users.

The time argument can have different formats. First, it can be an absolute time in the format hh:mm, in which hh is the hour (1 or 2 digits) and mm is the minute of the hour (in two digits). Second, it can be in the format +m, in which m is the number of minutes to wait. The word now is an alias for +0.

The -f flag means 'reboot fast'. This only creates an advisory file /fastboot which can be tested by the system when it comes up again.

The boot rc file can test if this file is present, and decide not to run fsck(1) since the system has been shut down in the proper way.

After that, the boot process should remove /fast- boot.

The -F flag means 'force fsck'. This only creates an advisory file /forcefsck which can be tested by the system when it comes up again.

The boot rc file can test if this file is present, and decide to run fsck(1) with a special 'force' flag so that even properly unmounted filesystems get checked. After that, the boot process should remove /forcefsck.

The -n flag causes shutdown not to call init, but to kill all running processes itself. shutdown will then turn off quota, accounting, and swapping and unmount all filesystems.

ACCESS CONTROL

shutdown can be called from init(8) when the magic keys CTRL-ALT- DEL are pressed, by creating an appropriate entry in /etc/inittab. This means that everyone who has physical access to the console keyboard can shut the system down. To prevent this, shutdown can check to see if an authorized user is logged in on one of the virtual consoles. If shutdown is called from init(8), it checks to see if the file /etc/shutdown.allow is present. It then compares the login names in that file with the list of people that are logged in on a virtual console (from /var/run/utmp). Only if one of those authorized users or root is logged in, it will proceed. Otherwise it will write the message shutdown: no authorized users logged in to the (physical) system console. The format of /etc/shutdown.allow is one user name per line. Empty lines and comment lines (prefixed by a #) are allowed. Currently there is a limit of 32 users in this file.

FILES

/fastboot /etc/inittab /etc/init.d/halt /etc/init.d/reboot /etc/shutdown.allow

BUGS

Not really a bug, but most users forget to give the time argument and are then puzzled by the error message shutdown produces. The time argument is mandatory; in 90 percent of all cases this argument will be the word now.

AUTHOR

Miquel van Smoorenburg, miquels@cistron.nl

SEE ALSO

fsck(8), init(1), halt(8), reboot(8)

EXAMPLES

su

NAME

su - run a shell with substitute user and group IDs

SYNOPSIS

su [-flmp] [-c command] [-s shell] [–login] [–fast] [–preserve-environment] [–command=command] [–shell=shell] [-] [–help] [–version] [user [arg...]]

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source.

This manual page documents the GNU version of su. su allows one user to temporarily become another user. It runs a shell with the real and effective user ID, group ID, and supplemental groups of USER. If no USER is given, the default is root, the super-user. The shell run is taken from USER's password entry, or /bin/sh if none is specified there. If USER has a password, su prompts for the password unless run by a user with real user ID 0 (the super-user). By default, su does not change the current directory. It sets the environment variables 'HOME' and 'SHELL' from the password entry for USER, and if USER is not the super- user, sets 'USER' and 'LOGNAME' to USER. By default, the shell is not a login shell. If one or more ARGs are given, they are passed as additional arguments to the shell. su does not handle /bin/sh or other shells specially (setting argv[0] to "-su", passing -c only to certain shells, etc.). On systems that have syslog, su can be compiled to report failed, and optionally successful, su attempts using sys- log.

OPTIONS

-c COMMAND, –command=COMMAND Pass COMMAND, a single command line to run, to the shell with a -c option instead of starting an interactive shell.

-f, –fast Pass the -f option to the shell. This probably only makes sense with csh and tcsh, for which the -f option prevents reading the startup file (.cshrc). With Bourne-like shells, the -f option disables filename pattern expansion, which is not a generally desirable thing to do.

–help Print a usage message on standard output and exit successfully.

-, -l, –login Make the shell a login shell. This means the following. Unset all environment variables except 'TERM', 'HOME', and 'SHELL' (which are set as described above), and 'USER' and 'LOGNAME' (which are set, even for the super-user, as described above), and set 'PATH' to a compiled-in default value. Change to USER's home directory. Prepend "-" to the shell's name, to make it read its login startup file(s).

-m, -p, –preserve-environment Do not change the environment variables 'HOME', 'USER', 'LOGNAME', or 'SHELL'. Run the shell given in the environment variable 'SHELL' instead of USER's shell from /etc/passwd, unless the user running su is not the superuser and USER's shell is restricted. A restricted shell is one that is not listed in the file /etc/shells, or in a compiled-in list if that file does not exist. Parts of what this option does can be overridden by –login and –shell.

-s, –shell shell Run SHELL instead of USER's shell from /etc/passwd, unless the user running su is not the superuser and USER's shell is restricted.

–version Print version information on standard output then exit successfully.

EXAMPLES

tail

NAME

tail - output the last part of files

SYNOPSIS

tail [-c [+]N[bkm]] [-n [+]N] [-fqv] [–bytes=[+]N[bkm]] [–lines=[+]N] [–follow] [–quiet] [–silent] [–verbose] [–help] [–version] [file...] tail [{-,+}Nbcfklmqv] [file...]

DESCRIPTION

This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source.

This manual page documents the GNU version of tail. tail prints the last part (10 lines by default) of each given file; it reads from standard input if no files are given or when a filename of '-' is encountered. If more than one file is given, it prints a header consisting of the file's name enclosed in '==>' and '<==' before the output for each file.

The GNU tail can output any amount of data, unlike the Unix version, which uses a fixed size buffer. It has no -r option (print in reverse). Reversing a file is really a different job from printing the end of a file; the BSD tail can only reverse files that are at most as large as its buffer, which is typically 32k. A reliable and more versatile way to reverse files is the GNU tac command.

OPTIONS

tail accepts two option formats: the new one, in which numbers are arguments to the option letters, and the old one, in which a '+' or '-' and optional number precede any option letters.

If a number ('N') starts with a '+', tail begins printing with the Nth item from the start of each file, instead of from the end.

-c N, –bytes N Tail by N bytes. N is a nonzero integer, optionally followed by one of the following characters to specify a different unit.

b 512-byte blocks.

k 1-kilobyte blocks.

m 1-megabyte blocks.

-f, –follow Loop forever trying to read more characters at the end of the file, on the assumption that the file is growing. Ignored if reading from a pipe. If more than one file is given, tail prints a header when- ever it gets output from a different file, to indicate which file that output is from.

-l, -n N, –lines N Tail by N lines. -l is only recognized using the old option format.

-q, –quiet, –silent Never print filename headers.

-v, –verbose Always print filename headers.

–help Print a usage message and exit with a status code indicating success.

–version Print version information on standard output then exit.

EXAMPLES

Let's watch syslog:

```
tail -f /var/log/messages
```

Checking on mail deliveries

```
tail -f /var/log/maillog
```

tar

NAME

tar - The GNU version of the tar archiving utility

SYNOPSIS

tar [-] A –catenate –concatenate | c –create | d –diff –compare | r –append | t –list | u –update | x -extract –get [–atime-preserve] [-b, –block-size N] [-B, –read-full-blocks] [-C, –directory DIR] [–checkpoint] [-f, –file [HOST-NAME:]F] [–force-local] [-F, –info-script F –new-volume- script F] [-G, –incremental] [-g, –listed-incremental F] [-h, –dereference] [-i, –ignore-zeros] [–ignore-failed-read] [-k, –keep-old-files] [-K, –starting-file F] [-l, –one-file-system] [-L, –tape-length N] [-m, –modification-time] [-M, –multi-volume] [-N, –after-date DATE, –newer DATE] [-o, –old-archive, –portability] [-O, –to-stdout] [-p, –same- permissions, –preserve-permissions] [-P, –absolute-paths] [–preserve] [-R, –record-number] [–remove-files] [-s, –same-order, –preserve-order] [–same-owner] [-S, –sparse] [-T, –files-from F] [–null] [–totals] [-v, –ver- bose] [-V, –label NAME] [–version] [-w, –interactive, –confirmation] [-W, –verify] [–exclude FILE] [-X, –exclude-from FILE] [-Z, –compress, –uncompress] [-z,–gzip, –ungzip] [–use-compress-program PROG] [–block- compress] [-[0-7][lmh]] filename1 [filename2, ... filenameN] directory1 [directory2, ...directoryN]

DESCRIPTION

This manual page documents the GNU version of tar , an archiving program designed to store and extract files from an archive file known as a tarfile. A tarfile may be made on a tape drive, however, it is also common to write a tarfile to a normal file. The first argument to tar must be one of the options: Acdrtux, followed by any optional functions. The final arguments to tar are the names of the files or directories which should be archived. The use of a directory name always implies that the subdirectories below should be included in the archive.

FUNCTION LETTERS

One of the following options must be used:

-A, –catenate, –concatenate append tar files to an archive

-c, –create create a new archive

-d, –diff, –compare find differences between archive and file system

–delete delete from the archive (not for use on mag tapes!)

-r, –append append files to the end of an archive

-t, –list list the contents of an archive

-u, –update only append files that are newer than copy in archive

-x, –extract, –get extract files from an archive

OTHER OPTIONS

–**atime-preserve** don't change access times on dumped files

-b, –**block-size** N block size of Nx512 bytes (default N=20)

-B, –**read-full-blocks** reblock as we read (for reading 4.2BSD pipes)

-C, –**directory** DIR change to directory DIR

–**checkpoint** print directory names while reading the archive

-f, –**file** [HOSTNAME:]F use archive file or device F (default /dev/rmt0)

–**force-local** archive file is local even if has a colon

-F, –**info-script F** –**new-volume-script** F run script at end of each tape (implies -M)

-G, –**incremental** create/list/extract old GNU-format incremental backup

-g, –**listed-incremental** F create/list/extract new GNU-format incremental backup

-h, –**dereference** don't dump symlinks; dump the files they point to

-i, –**ignore-zeros** ignore blocks of zeros in archive (normally mean EOF)

–**ignore-failed-read** don't exit with non-zero status on unreadable files

-k, –**keep-old-files** keep existing files; don't overwrite them from archive

-K, –**starting-file** F begin at file F in the archive

-l, –**one-file-system** stay in local file system when creating an archive

-L, –**tape-length** N change tapes after writing N*1024 bytes

-m, –**modification-time** don't extract file modified time

-M, –**multi-volume** create/list/extract multi-volume archive

-N, –**after-date DATE,** –**newer DATE** only store files newer than DATE

-o, –**old-archive,** –**portability** write a V7 format archive, rather than ANSI format

-O, –**to-stdout** extract files to standard output

-p, –**same-permissions,** –**preserve-permissions** extract all protection information

-P, –**absolute-paths** don't strip leading '/'s from file names

–**preserve** like -p -s

-R, –**record-number** show record number within archive with each message

–**remove-files** remove files after adding them to the archive

-s, –**same-order,** –**preserve-order** list of names to extract is sorted to match archive

–**same-owner** create extracted files with the same ownership

-S, –**sparse** handle sparse files efficiently

-T, –**files-from** F get names to extract or create from file F

–**null -T** reads null-terminated names, disable -C

–**totals** print total bytes written with –create

-v, –**verbose** verbosely list files processed

-V, –**label** NAME create archive with volume name NAME

–**version** print tar program version number

-w, –**interactive,** –**confirmation** ask for confirmation for every action

-W, –**verify** attempt to verify the archive after writing it

–**exclude** FILE exclude file FILE

-X, –**exclude-from** FILE exclude files listed in FILE

-Z, –**compress,** –**uncompress** filter the archive through compress

-z, –**gzip,** –**ungzip** filter the archive through gzip

–**use-compress-program** PROG filter the archive through PROG (which must accepted)

–**block-compress** block the output of compression program for tapes

-[0-7][lmh] specify drive and density

EXAMPLES

To check a tar file contents:

```
# tar -tf foo.tar
```

If it's gzipped

```
# tar -tzf foo.tar.gz
```

To extract the contents

```
# tar -xzf foo.tar.gz
```

To create a tar file of all the files in the /etc directory

```
# tar -cf etc.tar /etc/*
```

To create a gzipped file of all the files and directories in /usr/local and see the files as they are placed in the archive

```
# tar -czvf usr_local.tgz /usr/local/*
```

which

NAME

which - show full path of commands

SYNOPSIS

which progname ...

DESCRIPTION

Which takes a series of program names, and prints out the full pathname of the program that the shell would call to execute it. It does this by simulating the shells searching of the $PATH environment variable.

EXAMPLES

Shortcuts

Tab Completion

One of the nicest things about Linux is the tab completion. Tab completion allows you to type in the first few letters of a command or directory and then <Tab> to complete. This can save quite a few keystrokes when entering deep

directories with long names. All that is necessary are the first unique characters of the filename followed by a <Tab> and the sytem will complete. If you haven't entered enough characters, hitting <Tab><Tab> will provide all possible matches. Commands can also be completed with <Tab>. If you hit <Tab><Tab> with no arguments you will get something like this:

```
# <Tab><Tab>
There are 1661 possibilities.
Do you really wish to see them all? (y or n)
```

However if you enter just the first couple of letters like this:

```
# log <Tab><Tab>
logger logname logresolve logtail
login logout logrotate
```

Then you get all the possible commands with the first three letters log.

Filename Expansion

You can use wildcard characters to expand commands, to speed up access. Suppose you knew their was a configuration file, that you needed to modify, because you recently added some new libraries. You know the configuration file is in /etc and you know it begins with an l, but you can't remember the name. You could use the filename expansion capabilities of the shell to find ld.so.conf.

```
# cd /etc
# ls -d l*
ld.so.cache localtime@ logrotate.conf lynx.cfg
ld.so.conf localtime.rpmnew@ logrotate.d/
lilo.conf login.defs ltrace.conf
```

The "*" means substitute any or no string following the l. The -d argument says don't traverse directories. So the command returns all files and directories, in the current directory, that begin with the letter l. If we knew the file began with ld we might form the command

```
# ls -d ld*
ld.so.cache ld.so.conf
```

The "?" substitutes for any single character. It is more usefull for seperating similar filenames with single character differences. If we wanted to see a listing of all the files under /usr/man/ we could do the following;

```
# cd /usr/man
# ls
man1/ man3/ man5/ man7/ man9/ mann/
man2/ man4/ man6/ man8/ manl/ whatis
# ls man?
<returned_long_list>
```

Redirection

Redirection allows you to take the ouput from a command and redirect it to a file. This allows you to capture If we wanted to capture the output of the "ls man?"command we could redirect it to a file.

```
# ls man? > manfiles.list
# ls
man1/ man4/ man7/ manfiles.list whatis
man2/ man5/ man8/ manl/
man3/ man6/ man9/ mann/
```

If the file manfiles.list doesn't exist, it will be created. If there is a file manfiles.list it will be overwritten. It is also possible to redirect the input. We could mail the file we just created by redirecting the standard input of the mail command like this.

```
$ mail <manfiles.list
```

Suppose we want to append a list from another man directory /usr/local/man/ to this list. We would use the append ">>" redirection to add them to the manfiles.list like this.

```
# cd /usr/local/man
ls man? >> /usr/man/manfiles.list
```

Connecting with Pipes

It is possible to take the standard output of one command and direct it to the standard input of another command. This is known as piping. wc is a command the counts the number of words and characters. If we want to know how many words are in our manfiles.list, we can read the input of the wc command from the output of the cat command like this.

```
$ cat manfiles.list | wc
357 2111 14920
```

The symbol "|" is known as the "pipe" symbol. The standard output of the command on the left of the | becomes the input of the command to the right. We can string multiple commands together, to perform complex actions on data, using the pipe.

bash Environment

Bash is an sh-compatible command language interpreter that executes commands read from the standard input or from a file. Bash also incorporates useful features from the Korn and C shells (ksh and csh). Bash is the default shell for most Linux distributions. Generally speaking a shell is a user interface to the Linux operating system. It takes input from the user and interprets those command into instructions that the operating system can understand.

Bash is a character based interface. It takes it's instructions from standard input in textual form, and usually produces character based output.

The shell's job then is to interpret user input into commands understood by Linux. It breaks up the lines of the command into pieces, called words. It then interprets these words, according to rules. The shell sets up I/O according to the instructions in the commands. It finds the commands in the files and runs them with options given on the command line.

Each step is made of one or more substeps, each containing a particular instruction to the underlying operating system. The shell itself is not Linux, just the user interface to it. Unix is one of the first operating systems to make the user inerface independant of the operating system.

The shell can do more than just pass commands to the system for execution. The shell helps you use the computer. The shell is a command interpreter that can expand and change the commands according to built in rules. Of equal importance are wild-card and command connection operators that make the command line more general and flexible. The shell can take advantage of conditions in the environment, managed by the shell, to customise the way commands work.

Bash's command-line editing is one of the most powerful features. It's very easy to correct mistakes and modify previous commands to save time and keystrokes. Another major feature of bash is it's job control. Job control allows you to stop, start and pause any number of commands at the same time. You may wish to modify the characteristics of your environment, for this there are shell variables. A shell variable is a name with an associated value. Bash keeps track of several built-in variables. The convention is for shell variables to have names in all capital

letters. Bash-specific built-ins are, however defined in lowercase. The environmental variable will always be know by the subprocesses, however the shell needs to be told about other variables, options, aliases, etc., that are to be communicated to subprocesses. The way to do this is to declare them in the environment file. On Linux systems this is commonly the .bashrc file in the users home directory.

USERNAME

System login name.

EDITOR

Pathname to your text editor

HISTFILE

The name of the file, where the bash history is kept. Typically this is set to ~/.bash_history.

HISTSIZE

The size of the history file. How much history is saved.

MAIL

Name of the file to check for incoming mail

HOME

Name of your home (login) directory

HOSTNAME

The system hostname.

LANG

The Language environment. What language are you using?

LD_LIBRARY_PATH

Where the shell is to look for shared libraries.

OLDPWD

Previous directory before the last cd command

PWD

Current directory

TERM

Type of terminal you are using. The variable TERM is vitaly important for using any screen editor and for third-party applications. The value of TERM must be a short character string with lowercase that appears as a filename in the terminfo database. The terminfo databse describes how to tell terminals to do certain commaon things like position the cursor on the screen, go into reverse video, scroll, insert text, and so on.

PATH

The command search path. The directories and search order should check for

SHELL

Path to the shell command.

PS1

The primary prompt string. The default value on Red Hat Linux is "\u@\h \$". Here's a table of values you can use to customize you're prompt.

\d The date in "Weekday Month Day" format

\h	The hostname
\n	A carriage return and line feed
\s	The name of the shell
\t	The current time in HH-MM-SS format
\u	The username of the current user
\w	The current working directory
\W	The basename of the current working directory
\#	The command number of the current command
\!	The history number of the current command
\$	If the effective UID is 0 print a #, otherwise print a $
\nnn	Character code in octal
\\	Print a backslash
\[	Begins a sequence of non-prining characters, such as terminal command sequences
\]	End of a sequence of non-printing characters

USER

The current user of the shell.

MANPATH

Path to the system man pages. Typical setting would be something like this:

```
joe@localhost $ echo $MANPATH
/usr/man:/usr/X11R6/man:/usr/lib/perl5/man:/usr/share/perl5/man/
```

OTHERS

A Few Words About $PATH

One of the most baffling concepts for new users of Linux is the concept of $PATH. $PATH is an environmental variable. Environmental variables can come from many sources, but most are set in the user's .bash_profile. Environmental variables like $PATH tell the shell where to look for executable programs and in genereal how to find things on the system. which shell to use, and other important information about how the user can and will interact with the shell. If a program doesn't exist in the $PATH the shell cannot find, and therefore cannot execute the program. A common question, among new users, is why a command is not executed, even though the user is in the same directory as the binary. The answer is, the shell knows nothing about binaries that are not in the $PATH, unless the full path to the binary is given on the command line. This can be accomplished with several methods. We'll demonstrate two. First give the full path to the executable.

```
$ pwd
/home/joe
$ /this_directory/that_directory/my_command
```

The shell will execute my_command, if the user has the proper permissions, because the user has given the shell the full path to the executable.

Or if the command is in the current directory.

```
$ pwd
/this_directory/that_directory
$ ./my_command
```

The "./" tells the shell to execute my_command in the current directory.

To find out what is in your $PATH do the following.

```
$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/joeuser/bin
```

The order of the directories in $PATH is significant. It is the order which the shell will search for commands. If there are commands in different directories, with the same name, the first command found will be executed. This is important to know, because it can work against you or you can use it to your advantage.

File and Directory Ownership and Permissions

Because Linux is a multiuser sytem, files can be created by individual users, and are then "owned" by by that user until they either delete them, or give them to another user. Each user belongs to one or more "groups" and can share files with other members of that group. Permission determine how users on the system can use files and directories.

read permissions determine who is able to look at a file

write permissions determine who is able to change a file

execute permissions detrmine who is able to execute a file, or run the program associated with the file.

On occassion you will need to change these attributes. For example when you write a shell or perl script you must change the permissions to make the script executable.

Write and execute permissions for directories go together. Generally, if you allow users to write files to a directory, you also allow them to remove files from that directory. There exists a mechanism to allow users to only delete files from a directory, that they wrote. (See discussion below of the save program text attribute on directories)

To facilitate file sharing, Linux has three levels of permission, user, group and others. The user is generally the person who created the file. The group is determined by the /etc/group file and is often created by an administrator to create colaborative groups, for working on things in a cooperative way, such as groups of programmers working on a single project.

To examine the attributes of files use ls -l like this:

```
joe@localhost $ ls -l
drwxr-xr-x 2 joe joe 1024 Mar 27 18:17 bin/
-rw-rw-rw- 1 joe joe 0 Mar 26 19:22 firstfile
-rw-rw-rw- 1 joe joe 0 Mar 26 19:22 fourfile
-rw-rw-rw- 1 joe joe 0 Mar 26 19:22 lastfile
-rw-rw-rw- 1 joe joe 0 Mar 26 19:22 onefile
-rw-rw-rw- 1 joe joe 29 Mar 27 01:37 thatfile
-rw-rw-rw- 1 joe joe 29 Mar 26 19:23 thisfile
-rw-rw-rw- 1 joe joe 0 Mar 26 19:22 thosefiles
-rw-rw-rw- 1 joe joe 0 Mar 26 19:22 threefile
-rw-rw-rw- 1 joe joe 0 Mar 26 19:22 twofile
```

Working from right to left you can see the attributes of the files. First on the right, is the name of the file or directory. Second from right is the date and time of creation. Third is the file size. The fourth column is the group and fifth is owner for the file. Then we come to the first column, which is the file attributes. Each file has three levels of permissions or modes, owner, group and user. Owner The owner is the the user who created the file. The chown command is used to change this.

Group Files can be grouped together for access under /etc/group for projects where group access is neccessary. For security the default setup in Red Hat is to create a group for each user.

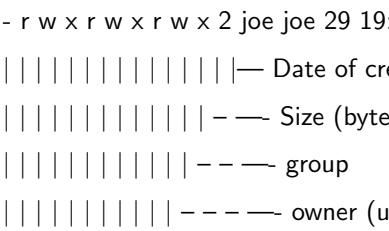
User Sometimes refered to as Public. All other users on the system. Anyone who has a username and can gain access to the system. There are three basic commands to manipulate these file attributes, in Linux. chown to change the ownership of the files. chgrp is used to change the group attributes. And chmod is used to change permissions or modes of files and directories.

There are three attributes for each level. The permissions have slightly different meaning in directories than they do for regualar files. Read If the read permission is set on a file, the user can look at the contents of the file. A user how has read permissions for a directory can list what files are in that directory. To get detailed information (ls -l) about the files in the directory, the execute permissions on the directory must also be set for that user. The read permissions on the files themselves determine if the user can read the contents of the files in the directory.

Write File write permissions determine whether the user will be allowed to modify the file. Write permissions on directories determine whether the user will be able to change files in the directory. That is whether the user will be able to add or remove files from the directory. The ability to remove files is determined by the permissions on the individual files in the directory.

Execute If the execute permission is set for the user, that user can use the file as a Linux system command. Execute permissions on a directory, means that a user can change directories to that directory and copy files from that directories, providing the read permissions are also set for that directory. The x permission for a directory is sometimes called the "search" permission.

By combining the three different levels of permissions and three user levels, there are nine different combinations of permisssions possible.



| | | | | | | | | | - - - - —- number of links
| | | | | | | | | - - - - - —- execute for public (other)
| | | | | | | | - - - - - - —- write for public (other)
| | | | | | | - - - - - - - —- read for public (other)
| | | | | | - - - - - - - - —- execute for group
| | | | | - - - - - - - - - —- write for group
| | | | - - - - - - - - - - —- read for group
| | | - - - - - - - - - - - —- execute for owner (user)
| | - - - - - - - - - - - - —- write for owner (user)
| - - - - - - - - - - - - - —- read for owner (user
- - - - - - - - - - - - - - —- suid & sticky bit & directory

The left three letters refer to the owner of the file, the middle three to the group, and the right three refer to the users (public). Missing permissions are refered to as “protection” and are indicated by a dash “-“. The nine protections are collectively the “mode” of the file or directory and can be changed wiht the chmod command.

Changing ownership

The ownership of files can be changed by root. This is done using the chown command.

root@localhost $ ls -l thisfile

-rw-rw-r– 1 joe joe 29 Mar 26 19:23 thisfile

root@localhost $ chown sam.sam thisfile

-rw-rw-r– 1 sam sam 29 Mar 26 19:23 thisfile

Changing permissions

The permissions or mode of a file or directory can be changed with the chmod command. This can only be done by the owner of the file, or super-user (root).

You cannot change a directory to a file, or a file to directory, with the chmod command.

There are several specifications that chmod accepts. The symbolic mode which uses the letters (ugoa), (+-=), (rwxXstugo). The letters ’ugoa’ control which users’ access will be changed. The owner (u), other users in the group (g), others not in the group (o), or all users (a). The ’+’ and ’-’ operators determine whether the permissions are add or removed, while the ’=’ operator causes the permissions to be the only ones set for the file.

root@localhost # ls -l thatfile

-rw-r–r– 1 joe joe 0 Mar 30 21:46 thatfile

root@localhost # chmod u+x thatfile

-rwxr–r– 1 joe joe 0 Mar 30 21:46 thatfile

The letters 'rwxXstugo' select the new permissions for the affected users:

read (r)

write (w)

execute (x) or access for directories

execute (X) only if the file is a directory or already has execute permissions for some users

suid (s) set user or group ID on execution, in other words, execute the program as the owner of the file instead of the user that called the program

save program text (t) "sticky bit" ignored by Linux on files, on directories only owners of file in that directory can delete it.

user (u) permissions that the user that is owner of the file currently has

group(g) permissions that the files group has for it.

others (o) permissions that others not user or group of the file have for it.

The numeric mode is derived by adding up the bites with values of 4 (r), 2 (w), and 1 (x). Omitted digits are assumed to be leading zeros.

Difficult unless you know binary arithmetic. Starting from left to right, each mode is represented as follows.

4000 is u+s

2000 is g+s

1000 is +t,

400 is r for user

200 is w for user

100 is x for user

40 is r for group

20 is w for group

10 is x for group

4 is r for other

2 is w for other

1 is x for other

So by adding up these number we can set the mode for a file, with a numerical value. Here's a simple example.

root@localhost # ls -l thisfile

-rw-rw-r– 1 joe joe 29 Mar 26 19:23 thisfile

root@localhost # chmod 666 thisfile

-rw-rw-rw- 1 sam sam 29 Mar 26 19:23 thisfile

The octal representations may seem difficult at first. Once you understand the octal representations, you may find them the easiest to use. Here are some of the common one:

644 is -rw-r–r– the owner can read and write, everyone else can read only.

755 is -rwxr-xr-x the owner card read, write and execute, everybody else can read or execute it. For a directory this is equivlent to

644

711 is -rwx–x–x the owner can read, write and execute, everyone else can only execute it.

444 is -r–r–r– this is read only permission for everyone.

There is other information contained in the 10 characters at the beginning of a listing about special files for devices, sockets, symbols, links, etc.. However since this is primarily a text for beginners, it will not be covered here.

Processes

Linux is a true multi-tasking operating system, This means that it can run more than one task at a time on behalf of the user. In Linux terminology each of the multiple tasks is called a process. Each of these processes is running independantly, with it's own allocated resources. One of the first processes that the kernel spawns upon startup is the getty process, which listens for and responds to, a login. Each time you ask the shell to run a command, it forks a seperate process.

If you create a shell pipeline with the | operarator, each command component is a seperate process. The comand line $ cat thisfile | sort will generate two processes,

one for each command. These processes communicate through the pipe. Some commands are internal to the shell and don't create any new processes.

A major purpose of the kernel is to provide control and support of the many programs that might want to use Central Processing Unit (CPU) at any one time. Several programs, or users may request access to the CPU at the same time. The kernel must grant access to only one application at a time. A process executes for only a short time, and then control passes to another process. This is why multi-tasking operating systems like Linux are called time sharing systems.

X is a process, which then in turn forks a new process when you call another program, or open an xterm. If you call a command in the xterm, this again forks another process, the command you asked to execute. This forking means that a new program is begun, with it's own process id (PID), while the calling program continues, ready to execute more commands from the user.

The ps command is used to see what processes are running. Output from the command typically looks something like this.

```
# ps
PID TTY STAT TIME COMMAND
639 p2 R 0:00 ps
16910 2 S 0:00 (mingetty)
17076 3 SW 0:00 (mingetty)
17077 4 SW 0:00 (mingetty)
17078 5 SW 0:00 (mingetty)
17079 6 SW 0:00 (mingetty)
26549 1 S 0:00 xterm -sb
28490 1 S 0:00 xterm -sb
28496 p2 S 0:00 su -
28497 p2 S 0:00 -bash
```

The output of the ps command is interpreted as follows:

PID the process id number of the program

TTY the terminal or pseudo-terminal that the program is being run for.

STAT information about the status of the process. R for runnable

S for sleeping or suspend, T for stopped or traced, Z for a zombie process, W if the process has no resident pages, and N if the process has a positive nice value

TIME

COMMAND The name of the process.

Background processes

The getty process (mingetty in the above example) is a "foreground" process, so called because it is in the foreground of your attention. Normally each command, or pipleline of commands is in the foreground. meaning they run, and you wait for them to return their results. It is also possible to execute programs as background processes. This means that you can execute the program, while still having your shell available for running other commands. The shell provides the & (ampersand) operator to allow you to run commands in the background. You simply add it to the end of a command line.

```
$ updatedb &
[1] 865
$
```

The command has not completed. It is still running, but you can use your system for other things while the backgrounded command proceeds. (updatedb updates the locate database.) When you run a command with the &, you get the shell prompt immediately and are able to enter further commands, even though the process you created is still running. The shell will return a process id (PID), so you can refer to that process, and then return your prompt for another command. Output will still come to your screen, because you did not redirect it. This can cause some confusion, since the output of the backgrounded process will be be mixed with the output of whatever command you are currently executing in the foreground.

Obviously the proper choice it to redirect the output of a background process to a file. You can also redirect the error output, so that it doesn't "pop up" on your screen.

System Startup and Initialization

Running the System

Why you shouldn't be root.

Most modern Linux distribution prompt you during the installation to create a user on the system, with good reason.

The root user is all powerful on a Linux system. With this power comes responsibility. Since Linux has not "trashbin" and no simple means to recover deleted files, when you tell it to remove files, it usually does so without hesitation and often without prompting you, unless there is an error. For the beginner, the lack of response from the system can often be disconcerting. If you are running as root, it can be hazardous to your system.

Many system commands have recursive options as well as wild card arguments that can create havoc on your system, if you are root. If you run as a regular user, you can still destroy a lot of data in your home directory, but you usually don't have the proper ownership and permissions to do much damage to the system as a whole. One of the reasons Linux is highly resistant to virus and worm infection is that most users on the system don't have root privileges, and therefore can't propogate virus and worms to system files. Operating as root would circumvent these protections and allow viruses and worms to spread through Linux systems much as they do in other operating systems.

Booting the System

In this section we cover various methods for booting the system, go into detail about the initialization system for Linux, and a brief overview of managing runlevels. We show you how to properly shutdown and reboot your system.

Boot floppy

Did you make a boot floppy during your install? Most distributions of Linux give you the opportunity to create a boot floppy during your install. This is always a good idea. Should you make a mistake like forgetting to run lilo after modifying /etc/lilo.conf a boot disk would allow you to recover and get your system back in running order.

Have you been running as root against our better advice, and removed part of your filesystem, where your kernel is? Have you installed a new kernel, and fogotten to save your old running kernel as a fallback position? All these scenarios are easier to recover from with a boot floppy.

Basically, a boot floppy, created during the install, contains a kernel, modules and ram disk that will work on your system. It contains many of the same configuration information gathered during the install. If you didn't create one during the install, don't fret, you can still create one now.

mkbootdisk

Mkbootdisk is a program that allows you to create a stand-alone boot floppy from a running system. (man mkbootdisk) It's the same script that the installer uses to create one during the install. The resulting boot disk is entirely self contained, and includes an initial ramdisk image which loads any necessary SCSI modules for the system. The created boot disk looks for the root filesystem on the device suggested by /etc/fstab. The only required argument is the kernel version to put onto the boot floppy.

You can create a boot floppy from the installed kernel, or from a newly compiled kernel. Some people prefer to use a boot floppy for testing a new kernel, instead of loading it from the bootloader. Here's a sample command line for creating a boot floppy on a newly installed Red Hat Linux system.

```
# mkbootdisk –device /dev/fd0 2.4.20-8
```

By default mkbootdisk give very little information, except for errors. For more information choose the –verbose argment like this

```
# mkbootdisk –device /dev/fd0 –verbose 2.4.20-8
```

If you ran mkbootdisk without errors, you can see what's on it by mounting the floppy.

```
# mount -t vfat /dev/fd0 /mnt/floppy
# ls -l /mnt/floppy
```

```
total 1346
-rwxr-xr-x 1 root root 208 Sep 23 19:16 boot.msg
-rwxr-xr-x 1 root root 246298 Sep 22 01:24 initrd.img
-r-xr-xr-x 1 root root 7836 Sep 23 19:16 ldlinux.sys
-rwxr-xr-x 1 root root 143 Sep 23 19:16 syslinux.cfg
-rwxr-xr-x 1 root root 1122186 Mar 13 2003 vmlinuz
```

So what are all these files?

```
# cat /mnt/floppy/boot.msg
Press <return> (or wait 10 seconds) to boot your Red Hat Linux
system from /dev/rd/c0d0p3. You may override the default linux
kernel parameters by typing "linux <params>", followed by <re-
turn> if you like.
# cat /mnt/floppy/syslinux.cfg
default linux
prompt 1
display boot.msg
timeout 100
label linux
      kernel vmlinuz
    append initrd=initrd.img ro hdc=ide-scsi root=/dev/rd/c0d0p3
```

syslinux.cfg is the configuration file for the bootloader to instruct it what then name of the ram disk is, that hdc requires the ide-scsi module and where the root file system is on the disk.

So boot.msg is the message you get when you boot from the floppy. initrd.img is the ram disk that contains the temporary root file system, with kernel modules and other information required to boot the system. ldlinux.sys is the system loader.

Now just to make sure, you should boot from the floppy and make sure it works properly. Once you've tested it, label it, write protect it and put it in a safe place. It's good insurance, and like all good insurance, we hope you'll never need it, but you'll be glad you have it when you need it.

LILO

The majority of this information came directly from the documentation installed on the system during installation check /usr/share/doc/lilo-xx.x.x/doc/User_Guide.ps for a more detailed discussion of the finer points. The Xs following lilo represent the specific version on your system. If you are running X on your system you can view these documents with gv. To read them on your system go Menu -> Graphic Programs and find ghostview. Once you have it running File -> Open and drill down to /usr/share/doc and find the file User_Guide.ps. You can also find informtion in the man pages for lilo(8) and lilo.conf(5).

LILO is a versatile boot loader for Linux. It does not depend upon a specific file system, can boot Linux images from a floppy disk and from hard disks and act as a "boot manager" for other operating systems.

Up to sixteen different images can be selcected at boot time. Various paramters can be set independently for each kernel. LILO can even be used as the master boot record.

What is lilo

LILO is a collection of several programs and other files.

The map installer is the program you run under Linux to put all files belonging to LILO at the appropriate places and to record information about the location of data needed at boot time. This program normally resides in /sbin/lilo. It has to be run to refresh that information whenever any part of the system that LILO knows about changes, e.g. after installing a new kernel.

Various files contain data LILO needs at boot time, e.g. the boot loader. Those files normally reside in /boot. The most important files are the boot loader (see below) and the map file (/boot/map), where the map installer records the location of the kernel(s).[*] Another important file is the configuration file, which is normally called /etc/lilo.conf

The boot loader is the part of LILO that is loaded by the BIOS and that loads kernels or the boot sectors of other operating systems. It also provides a simple command-line interface to interactively select the item to boot and to add boot options.

LILO primarily accesses the following parts of the system:

The root file system partition is important for two reasons: first, LILO sometimes has to tell the kernel where to look for it. Second, it is frequently a convenient place for many other items LILO uses, such as the boot sector, the /boot directory, and the kernels.

The boot sector contains the first part of LILO's boot loader. It loads the much larger second-stage loader. Both loaders are typically stored in the file /boot/boot.b

The kernel is loaded and started by the boot loader. Kernels typically reside in the root directory or in /boot.

Booting Basics

When booting from a floppy disk, the first sector of the disk, the so-called boot sector, is loaded. That boot sector contains a small program that loads the respective operating system. MS-DOS boot sectors also contain a data area, where disk and file system parameters (cluster size, number of sectors, number of heads, etc.) are stored.

When booting from a hard disk, the very first sector of that disk, the so-called master boot record (MBR) is loaded. This sector contains a loader program and the partition table of the disk. The loader program usually loads the boot sector, as if the system was booting from a floppy.

Note that there is no functional difference between the MBR and the boot sector other than that the MBR contains the partition information but doesn't contain any file system-specific information (e.g. MS-DOS disk parameters).

The first 446 (0x1BE) bytes of the MBR are used by the loader program. They are followed by the partition table, with a length of 64 (0x40) bytes. The last two bytes contain a magic number that is sometimes used to verify that a given sector really is a boot sector.

LILO in the MBR

This is a "safe" LILO setup: LILO is booted by the DOS-MBR. No other boot sectors have to be touched. If the other OS (or one of them, if there are several other operating systems being used) should be booted without using LILO, the other partition has to be marked "active" with fdisk or activate.

Installation:

- install LILO with its boot sector on the Linux partition.
- use fdisk or activate to make that partition active.
- reboot.

When you choose LILO as your boot loader during the installation of Linux and you choose to install it into the MBR this is essentially what the installer does.

Bios Limitations

The most common BIOS restrictions that affect LILO are the limitation to two hard disks and the inability to access more than 1024 cylinders per disk. LILO can detect both conditions, but in order to work around the underlying problems, manual intervention is necessary.

The drive limit does not exist in every BIOS. Some modern motherboards and disk controllers are equipped with a BIOS that supports more (typically four) disk drives. When attempting to access the third, fourth, etc. drive, LILO prints a warning message but continues. Unless the BIOS really supports more than two drives, the system will not be able to boot in that case

There are four approaches of how such problems can be solved:

- use of a different partition which is on an accessible disk and which does not exceed the 1024 cylinder limit. If there is only a DOS partition which fulfills all the criteria, that partition can be used to store the relevant files.
- rearranging partitions and disks. This is typically a destructive operation, so extra care should be taken to make good backups.
- if the system is running DOS or Windows 95, LOADLIN can be used instead of LILO.
- if all else fails, installation of a more capable BIOS, a different controller or a different disk configuration.

LILO depends on the BIOS to load the following items:

- /boot/boot.b
- /boot/map (created when running /sbin/lilo)
- all kernels
- the boot sectors of all other operating systems it boots
- the startup message, if one has been defined

Typically, what most Linux distributions do, is create a small /boot partition at the beginning of the drive from 50-100Mb. This insures that the required files will be within the 1024 cylinder boundary. Other strategies include using Logical Block Addressing (LBA), which “fools” the bios into seeing the disc geometry with 1024 cylinders or less.

The boot prompt

Immediately after it's loaded, LILO checks whether one of the following is happening:

- any of the [Shift], [Control] or [Alt] keys is pressed.
- [CapsLock] or [ScrollLock] is set.

If this is the case, LILO displays the "boot:" prompt and waits for the name of a boot image (i.e. Linux kernel or other operating system). Otherwise, it boots the default boot image or - if a delay has been specified - waits for one of the listed activities until that amount of time has passed.

At the boot prompt, the name of the image to boot can be entered. Typing errors can be corrected with [BackSpace], [Delete], [Ctrl U] and [Ctrl X]. A list of known images can be obtained by pressing [?] or [Tab].

If [Enter] is pressed and no file name has been entered, the default image is booted.

Boot command-line options

LILO is also able to pass command-line options to the kernel. Command-line options are words that follow the name of the boot image and that are separated by spaces.

```
boot: linux single root=200
```

Recent kernels recognize a large number of options, among them are debug, no387, no-hlt, ramdisk=size, reserve=base,size,, root=device, ro, and rw. All current init programs also recognize the option single. The options lock and vga are processed by the boot loader itself. Boot command-line options are always case-sensitive.

There is also a plethora of options to specify certain characteristics (e.g. IO and memory addresses) of devices. Some common ones are ether, floppy, hd, bmouse, and sound. The usage of these options is option=number,. Please consult the corresponding FAQs and HOWTOs for details.

Options of the type variable=value which are neither standard options nor device-specific options, cause the respective variables to be set in the environment passed to init. The case of the variable name is preserved, i.e. it isn't automatically converted to upper case.

See the section on init below for further clues about these options.

GRUB

Introduction

This information is summarized from the GNU/GRUB web site http://www.gnu.org/software/grub/#docu and from the grub info pages ("info grub") Many Linux distributions give you the option of using GRUB as you bootloader, during the install. You don't need all this information to use grub. However, if you need to dual boot to a proprietary OS, or need to recover from a damaged Master Boot Record, you may find this information valuable.

GNU GRUB is a Multiboot boot loader. It was derived from GRUB, GRand Unified Bootloader, which was originally designed and implemented by Erich Stefan Boleyn.

Briefly, boot loader is the first software program that runs when a computer starts. It is responsible for loading and transferring control to the operating system kernel software (such as the Hurd or the Linux). The kernel, in turn, initializes the rest of the operating system (e.g. GNU).

GNU GRUB follows these requirements below:

- Compliant with the Multiboot Specification.
- Basic functions are easy for an end-user to use.
- Rich functionality for OS experts/designers.
- Compatibility for booting FreeBSD, NetBSD, OpenBSD, and GNU/Linux. Proprietary OS's such as Windows 9x/NT/2000/XP, and OS/2 are supported via a chain-loading function.

In addition to the requirements above, GNU GRUB has the following features:

- Support multiple executable formats.
- Support non-Multiboot OS's.
- Load multiple modules.
- Support a human-readable configuration file.
- Menu interface.
- Flexible command-line interface.
- Support multiple filesystem types.
- Support automatic decompression.

- Access data on any installed device.
- Geometry translation independent.
- Detect all installed RAM.
- Support Logical Block Address (LBA) mode.
- Download OS images from a network.
- Support diskless systems.
- Support remote terminals.

Sounds like the swiss army knife of boot loaders!

One of the important features in GRUB is flexibility; GRUB understands filesystems and kernel executable formats, so you can load an arbitrary operating system the way you like, without recording the physical position of your kernel on the disk.

Thus you can load the kernel just by specifying its file name and the drive (and the partition) where the kernel resides. To let GRUB know the drive and the file name, you can either type in them manually via the command-line interface, or use the nice menu interface through which you can easily select which OS it boots. To allow you to customize the menu, GRUB will load a preexisting configuration file . Note that you can not only enter the command-line interface whenever you like, but also you can edit specific menu entries prior to using them.

Naming conventions

The device syntax used in GRUB is a wee bit different from what you may have seen before in your operating system(s), and you need to know it so that you can specify a drive/partition.

(hd0,1)

Here, 'hd' means it is a hard disk drive. The first integer '0' indicates the drive number, that is, the first hard disk, while the second integer, '1', indicates the partition number (or the PC slice number in the BSD terminology). Once again, please note that the partition numbers are counted from _zero_, not from one. This expression means the second partition of the first hard disk drive. In this case, GRUB uses one partition of the disk, instead of the whole disk.

Of course, to actually access the disks or partitions with GRUB, you need to use the device specification in a command, like 'root (fd0)' or 'unhide (hd0,2)'. To help you find out which number is a partition you want, the GRUB command-line options have argument completion. That means that, for example, you only

need to type 'root (', followed by a <TAB>, and GRUB will display the list of drives, partitions, or file names, so it should be quite easy to determine the name of your target partition, even with minimal knowledge of the syntax.

Note that GRUB does _not_ distinguish IDE from SCSI - it simply counts the drive numbers from zero, regardless of their type. Normally, any IDE drive number is less than any SCSI drive number, although that is not true if you change the boot sequence by swapping IDE and SCSI drives in your BIOS.

Now the question is, how to specify a file? Again, see this example:

(hd0,0)/vmlinuz

This specifies the file named 'vmlinuz', found on the first partition of the first hard disk drive.

How to boot operating systems

GRUB has two distinct boot methods. One of the two is to load an operating system directly, and the other is to chain-load another boot loader which then will load an operating system actually. Generally speaking, the former is desirable, because you don't need to install or maintain other boot loaders and GRUB is flexible enough to load an operating system from an arbitrary disk/partition. However, the latter is sometimes required, since GRUB doesn't support all the existing operating systems natively.

Loading an operating system directly

Multiboot is the native format supported by GRUB. For the sake of convenience, there are also support for Linux, FreeBSD, NetBSD and OpenBSD. If you want to boot other operating systems, you will have to chain-load them.

Generally, GRUB can boot any Multiboot-compliant OS in the following steps:

1. Set GRUB's root device to the drive where the OS images are stored by the command 'root'
2. Load the kernel image by the command 'kernel' .
3. If you need modules, load them with the command 'module' or 'module-nounzip'.
4. Run the command 'boot'

Linux can be booted in a similar manner. You can load a kernel image by the command 'kernel' and then run the command 'boot'. If the kernel requires some parameters, just append the parameters to 'kernel', after the file name of the kernel.

It is relatively easy to boot Linux from GRUB, because it somewhat resembles to boot a Multiboot-compliant OS.

1. Set GRUB's root device to the same drive as Linux's. Probably the command 'find /vmlinuz' or similar can help you.
2. Load the kernel:

   grub> kernel /vmlinuz root=/dev/hda1

If you need to specify some kernel parameters, just append them to the command. For example, to set 'vga' to 'ext', do this:

grub> kernel /vmlinuz root=/dev/hda1 vga=ext

Configuration

You probably noticed that you need to type several commands to boot your OS. There's a solution to that - GRUB provides a menu interface from which you can select an item (using arrow keys) that will do everything to boot an OS.

To enable the menu, you need a configuration file, 'grub.conf' under the boot directory. We'll analyze an example file.

The file first contains some general settings, the menu interface related options. You can put these commands before any of the items (starting with 'title').

#

Sample boot menu configuration file

#

As you may have guessed, these lines are comments. Lines starting with a hash character ('#'), and blank lines, are ignored by GRUB.

By default, boot the first entry. default 0

The first entry (here, counting starts with number zero, not one!) will be the default choice.

Boot automatically after 30 secs. timeout 30

As the comment says, GRUB will boot automatically in 30 seconds, unless interrupted with a keypress.

Fallback to the second entry. fallback 1

If, for any reason, the default entry doesn't work, fall back to the second one (this is rarely used, for obvious reasons).

Now, on to the actual OS definitions. You will see that each entry begins with a special command, 'title', and the action is described after it. Note that there is no command 'boot' at the end of each item. That is because GRUB automatically executes 'boot' if it loads other commands successfully.

The argument for the command 'title' is used to display a short title/description of the entry in the menu. Since 'title' displays the argument as is, you can write basically anything in there.

```
# For booting Linux
title Linux kernel (hd0,0)/vmlinuz root=/dev/hda1
```

This boots Linux from the first hard disk.

```
# For booting Windows NT or Windows95
title Windows NT / Windows 95 boot menu
root (hd0,0)
makeactive
chainloader +1
# For loading DOS if Windows NT is installed
# chainload /bootsect.dos
```

This boots Windows.

Creating a GRUB boot floppy

To create a GRUB boot floppy, you need to take the files 'stage1' and 'stage2' from the image directory, and write them to the first and the second block of the floppy disk, respectively.

Caution: This procedure will destroy any data currently stored on the floppy.

On a Linux, that is done with the following commands:

```
# cd /usr/share/grub/i386-pc
# dd if=stage1 of=/dev/fd0 bs=512 count=1
1+0 records in
1+0 records out
# dd if=stage2 of=/dev/fd0 bs=512 seek=1
153+1 records in
153+1 records out
#
```

A GRUB boot floppy is a great recovery tool.

The grub shell

You can use the command 'grub' for installing GRUB under your operating systems and for a testbed when you add a new feature into GRUB or when fix a bug. 'grub' is almost the same as the Stage 2, and, in fact, it shares the source code with the Stage 2 and you can use the same commands in 'grub'. It is emulated by replacing BIOS calls with UNIX system calls and libc functions.

The command 'grub' accepts the following options:

–**help** Print a summary of the command-line options and exit.

–**version** Print the version number of GRUB and exit.

–**verbose** Print some verbose messages for debugging purpose.

–**device-map=FILE** Use the device map file FILE. The format is described in.

–**no-floppy** Do not probe any floppy drive. This option has no effect if the option '–device-map' is specified

–**probe-second-floppy** Probe the second floppy drive. If this option is not specified, the grub shell does not probe it, as that sometimes takes a long time. If you specify the device map file, the grub shell just ignores this option.

–**config-file=FILE** Read the configuration file FILE instead of '/boot/grub/grub.conf'. The format is the same as the normal GRUB syntax. See *Note Filesystem::, for more information.

–**boot-drive=DRIVE** Set the stage2 BOOT_DRIVE to DRIVE. This argument should be an integer (decimal, octal or hexadecimal).

–**install-partition=PAR** Set the stage2 INSTALL_PARTITION to PAR. This argument should be an integer (decimal, octal or hexadecimal).

–**no-config-file** Do not use the configuration file even if it can be read.

–**no-curses** Do not use the curses interface even if it is available.

–**batch** This option has the same meaning as '–no-config-file –no-curses'.

–**read-only** Disable writing to any disk.

–**hold** Wait until a debugger will attach. This option is useful when you want to debug the startup code.

General commands Commands usable both in the menu and in the command-line.

bootp:: Initialize a network device via BOOTP

color:: Color the menu interface

device:: Specify a file as a drive

dhcp:: Initialize a network device via DHCP

hide:: Hide a partition

ifconfig:: Configure a network device manually

pager:: Change the state of the internal pager

partnew:: Make a primary partition

parttype:: Change the type of a partition

password:: Set a password for the menu interface

rarp:: Initialize a network device via RARP

serial:: Set up a serial device

setkey:: Configure the key map

splashimage:: Use a splash image

terminal:: Choose a terminal

terminfo:: Define escape sequences for a terminal

tftpserver:: Specify a TFTP server

unhide:: Unhide a partition

Menu commands These commands can only be used in the menu:

default:: Set the default entry

fallback:: Set the fallback entry

hiddenmenu:: Hide the menu interface

timeout:: Set the timeout

title:: Start a menu entry

Summary Grub is a powerful bootloader with a built in shell like language. It can be used to boot multiple Operating Systems, as well as being a recovery tool for systems that have damaged bootloaders. It's another tool every Linux user should become familiar with.

Specifying boot time options

During the boot process it is possible to pass arguments to the kernel in order to enable specific modules or modify the default configuration set in the boot loader configuration.

It is important to note that spaces should not be used in a boot argument, but only between separate arguments. A list of values that are for a single argument are to be separated with a comma between the values, and again without any spaces. For example:

ether=9,0x300,0xd0000,0xd4000,eth0 root=/dev/hda1 *RIGHT*

ether = 9, 0x300, 0xd0000, 0xd4000, eth0 root = /dev/hda1 *WRONG*

LILO is the most common boot loader on Linux systems. Typically the boot loader will print LILO: to the screen then wait a few seconds for input. With no input it will boot to the default system. Typical system labels that people use in the LILO configuration files are linux and Windows and msdos. If you want to pass arguments to the kernel, this is the place to do it. For example:

LILO: linux root=/dev/hda1

LILO comes with excellent documentation, as discussed above. The LILO append= command is usefull if you want to add a boot time argument as a permanent addition to the LILO config file. You simply add something like append = "hd=64,32,202" to the /etc/lilo.conf file, to pass paramters about hard drive geometry, for example. It can either be added at the top of the config file, making it apply to all sections, or to a single system section by adding it inside an image= section. Please see the LILO documentation for a more complete description.

How the kernel sorts arguments

Most boot arguments take the form:

name[=value_1][,value_2]...[,value_11]

"name" is a unique keyword that is used to identify what part of the kernel the associated values (if any) are to be given to. Multiple boot args are just a space separated list of the above format. There is a limit of 11 parameters that can be associated with each keyword, though you can reuse a keyword for additional parameters, if the setup function supports it.

First, the kernel checks to see if the argument is any of the special arguments 'root=', 'ro', 'rw', or 'debug'. Then it walks a list of setup functions (contained in the bootsetups array) to see if the specified argument string (such as 'foo') has been associated with a setup function (foo_setup()) for a particular device or part of the kernel. If you passed the kernel the line foo=3,4,5,6,bar then the kernel would search the bootsetups array to see if 'foo' was registered. If it was, then it would call the setup function associated with 'foo' (foo_setup()) and hand it the integer arguments 3, 4, 5 and 6 as given on the kernel command line, and also hand it the string argument bar.

Anything of the form 'foo=bar' that is not accepted as a setup function above is then interpreted as an environment variable to be set. An example would be to use TERM=vt100 or BOOT_IMAGE=vmlinuz.bak as a boot argument. These environment variables are typically tested for in the initialization scripts to enable or disable a wide range of things.

Any remaining arguments that were not picked up by the kernel and were not interpreted as environment variables are then passed onto process one, which is usually the init program. The most common argument that is passed to the init process is the word single which instructs init to boot the computer in single user mode, and not launch all the usual daemons.

The 'root=' Argument

This argument tells the kernel what device is to be used as the root filesystem while booting. The default of this setting is the value of the root device of the system that the kernel was built on or the parameters set during the installation in lilo.conf or grub.conf. For example, if the lilo.conf has 'root=/dev/hda1' in the stanza for that image, then the default root device would be '/dev/hda1'. To override this default value, and select the first partition on the second hard drive as the root device, one would use 'root=/dev/hdb1'.

Valid root devices are any of the following devices:

1. /dev/hdaN to /dev/hddN, which is partition N on ST-506 compatible disk 'a to d'.
2. /dev/sdaN to /dev/sdeN, which is partition N on SCSI compatible disk 'a to e'.
3. /dev/xdaN to /dev/xdbN, which is partition N on XT compatible disk 'a to b'.

4. /dev/fdN, which is floppy disk drive number N. Having N=0 would be the DOS 'A:' drive, and N=1 would be 'B:'.

5. /dev/nfs, which is not really a device, but rather a flag to tell the kernel to get the root fs via the network.

6. /dev/ram, which is the RAM disk.

The more awkward and less portable numeric specification of possible disk devices in major/minor format is also accepted. So, for example /dev/sda1 is major 8, minor 1, so you could use root=0x801 as an alternative.

Other paramers

Other parameters that can be passed to the kernel include:

rootflags= allows you to give options pertaining to the mounting of the root filesystem

rootfstype= allows you to give a comma separated list of fs types that will be tried for a match when trying to mount the root filesystem

ro tells the kernel to mount the root filesystem as 'readonly'

rw tells the kernel to mount the root filesystem as read/write

nfsroot= tells the kernel which machine, what directory and what NFS options to use for the root filesystem, root=/dev/nfs is required

ip= or nfsaddrs= sets up the various network interface addresses that are required to communicate over the network

ramdisk_start= allow a kernel image to reside on a floppy disk along with a compressed ramdisk image

load_ramdisk= tells the kernel whether it is to try to load a ramdisk image or not

prompt_ramdisk= tells the kernel whether or not to give you a prompt asking you to insert the floppy containing the ramdisk image

ramdisk_size= override the default of 4096 (4MB) to a bigger or smaller size with this boot argument

ramdisk_blocksize= tuned for better memory management behaviour

noinitrd determines what happens to the initrd data after the kernel has booted

cachesize= override level 2 CPU cache size detection (in kB)

mem= to specify the amount of installed memory (or a value less than that if you wanted to limit the amount of memory available to linux)

memfrac= memory is broken down into zones, the two (or three) integers supplied here determine how much memory in each zone should be kept free

swap= tune some of the virtual memory (VM) parameters that are related to swapping to disk

buff= allows the user to tune some of the parameters related to buffer memory management

acpi= currently this only accepts 'off' to disable the ACPI subsystem

console= to be able to use another device like a serial port (or even a printer!) to be the console when no video device is present

debug the kernel communicates important (and not-so important) messages to the operator via the printk() function

decnet= supply two comma separated integers here to give your area and node respectively.

devfs= if you are using devfs, instead of the standard static devices in /dev/ then you can supply the words only or mount with this argument

gpt if you are using EFI GUID Partition Table handling, you can use this to override problems associated with an invalid PMBR

idle= setting this to 'poll' causes the idle loop in the kernel to poll on the need reschedule flag instead of waiting for an interrupt to happen

init= if for example, your init program got corrupted and thus stopped you from being able to boot, you could simply use the boot prompt init=/bin/sh which would drop you directly into a shell at boot, allowing you to replace the corrupted program

isapnp= takes the form of: isapnp=read_port,reset,skip_pci_scan,verbose

isapnp_reserve_dma= takes the form of: isapnp_reserve_dma=n1,n2,n3,...nN where n1 ... nN are the DMA channel numbers to not use for PnP

isapnp_reserve_io= takes the form of: isapnp_reserve_irq=io1,size1,io2,size2,...ioN,sizeN where ioX,sizeX are I/O start and length pairs of regions in I/O space that are not to be used by PnP

isapnp_reserve_irq= takes the form of: isapnp_reserve_irq=n1,n2,n3,...nN where n1 ... nN are the interrupt numbers to not use for PnP

isapnp_reserve_mem= takes the form of: isapnp_reserve_mem=mem1,size1,mem2,size2,...memN,sizeN where ioX,sizeX are I/O start and length pairs of regions in memory space that are not to be used by PnP

kbd-reset forces a reset at initialization time

lockd.udpport= and lockd.tcpport tell the kernel to use the given port numbers for NFS lockd operation (for either UDP or TCP operation).

maxcpus= number given with this argument limits the maximum number of CPUs activated in SMP mode

max_scsi_luns= allows you to set the maximum number of probed LUNs

mca-pentium prevent lock up on the test to detect the type of math chip coupling on microchannel machines

md= tell the kernel the multiple device layout

nmi_watchdog= hecks to see if the interrupt count is increasing (indicating normal system activity) and if it is not then it assumes that a processor is stuck and forces an error dump of diagnostic information

no387 causes Linux to ignore the math coprocessor even if you have one

no-hlt run an infinite loop when there is nothing else to do, and to not halt your CPU when there is no activity

no-scroll disables scrolling features that make it difficult to use Braille terminals

noapic do not use some of the advanced features of the interrupt controller on multi processor machines

noht disable hyper-threading on intel processors that have this feature

noisapnp if ISA PnP is built into the kernel, this will disable it

nomce disable the ability in some newer processors to self-monitor and detect inconsistencies and create Machine Check Exception and halt the system

nosmp If software suspend is enabled, and a suspend to disk file has been specified, using this argument will give a normal boot and the suspend data will be ignored

notsc tell the kernel to not use the Time Stamp Counter for anything, even if the CPU has one

nofxsr tell the kernel to not use any speed-up tricks involving the floating point unit, even if the processor supports them

panic= automatically reset after a kernel panic so that the machine comes back on line

pci= argument (not avail. in v2.0 kernels) can be used to change the behaviour of PCI bus device probing and device behaviour.

pirq= tells a SMP kernel information on the PCI slot versus IRQ settings for SMP motherboards which are unknown (or known to be blacklisted)

profile= kernel developers can profile how and where the kernel is spending its CPU cycles in an effort to maximize efficiency and performance

quiet only important and system critical kernel messages are printed to the console. Normal messages about hardware detection at boot are suppressed

raid= accepts noautodetect at the moment. See also md=

reboot= controls the type of reboot that Linux will do when it resets the computer

reserve= used to protect I/O port regions from probes

resume= If you are using software suspend, then this will allow you to specify the file name of the suspend to disk data that you want the machine to resume from.

scsi_logging= turns on logging of all SCSI events (error, scan, mlqueue, mlcomplete, llqueue, llcomplete, hlqueue, hlcomplete)

st= boot time configuration of the SCSI tape driver

video= argument (not avail. in v2.0 kernels) is used when the frame buffer device abstraction layer is built into the kernel

vga= not really a boot argument, allows the setup code to use the video BIOS to change the default display mode before actually booting the Linux kernel

IDE Disk/CD-ROM Driver Parameters

The IDE driver accepts a number of parameters, which range from disk geometry specifications, to support for advanced or broken controller chips. The following is a summary of some of the more common boot arguments. For full details, you really should consult the file ide.txt in the kernel source tree linux/Documentation directory, from which this summary was extracted.

hdx= is recognized for all ”x” from ”a” to ”h”, such as ”hdc”.

idex= is recognized for all ”x” from ”0” to ”3”, such as ”ide1”.

hdx=noprobe drive may be present, but do not probe for it

hdx=none drive is NOT present, ignore cmos and do not probe

hdx=nowerr ignore the WRERR_STAT bit on this drive

hdx=cdrom drive is present, and is a cdrom drive

hdx=cyl,head,sect disk drive is present, with specified geometry

hdx=autotune driver will attempt to tune interface speed to the fastest PIO mode supported, if possible for this drive only. Not fully supported by all chipset types, and quite likely to cause trouble with older/odd IDE drives.

idex=noprobe do not attempt to access/use this interface

idex=base probe for an interface at the addr specified, where "base" is usually 0x1f0 or 0x170 and "ctl" is assumed to be "base"+0x206

idex=base,ctl specify both base and ctl

idex=base,ctl,irq specify base, ctl, and irq number

idex=autotune driver will attempt to tune interface speed to the fastest PIO mode supported, for all drives on this interface. Not fully supported by all chipset types, and quite likely to cause trouble with older/odd IDE drives.

idex=noautotune driver will NOT attempt to tune interface speed This is the default for most chipsets, except the cmd640.

idex=serialize do not overlap operations on idex and ide(xˆ1)

The following are valid ONLY on ide0, and the defaults for the base,ctl ports must not be altered.

ide0=dtc2278 probe/support DTC2278 interface

ide0=ht6560b probe/support HT6560B interface

ide0=cmd640_vlb *REQUIRED* for VLB cards with the CMD640 chip (not for PCI – automatically detected)

ide0=qd6580 probe/support qd6580 interface

ide0=ali14xx probe/support ali14xx chipsets (ALI M1439/M1445)

ide0=umc8672 probe/support umc8672 chipsets

Other Drivers

There are additional drivers and associated boot arguments for old MFM/RLL/Standard ST-506 disk drives, XT disk drives, sound devices, old CD-ROM drives, ISDN, serial devices, ethernet devices, floppy disk drives, bus mice, printer drivers and parallel ports. If you need information on those drivers, see the HOWTO referenced below.

For a more detailed discussion of this topic see the BootPrompt HOWTO http://www.ibiblio.org/mdw/HOWTO/BootPrompt-HOWTO.html

Single User Mode

The comman "linux single" during the boot put your computer into runlevel 1, or single-user mode. You will be logged in as root , and you're not supposed to enter any passwords as you normally are. A "backdoor" into Linux for you as root so that you have a chance to correct any errors that might occur.

Here are some examples:

Lets say that you run kdm, gdm or xdm via /etc/inittab. If you by accident make an error in the setup of X or inittab the result might be that you never get to the graphical login because the bootprocess never get that far. That's when "linux single" is there to help you. It boots Linux to a root prompt so that you can correct the errors and reboot the computer.

You're running as root (you know better don't you?), and you leave a process running that logs through syslog. It fills up your log file, and in turn your partition. Now you can't log in. In single user mode, you can fix it.

You forgot your root password and need to reset it.

If you are using LILO, at the LILO boot prompt (if you are using the graphical LILO, press [Ctrl]x to exit the graphical screen and go to the boot: prompt);

```
boot: linux single
```

The GRUB boot loader doesn't have a default boot prompt. You can choose "e" when the menu displays, to edit the boot paramters, select the line that starts with kernel, for the kernel you want to boot. Go to the end of the line and type single as a seperate word (press the [Spacebar] and then type single. Press Enter to exit edit mode. Back at the GRUB screen, type b to boot into single user mode. You might find it simpler to edit grub.conf and add a single user login menu item similar to this:

```
title Single User Mode
      root (hd0,0)
```

```
kernel /vmlinuz-2.4.20-8 single rw root=LABEL=/ hdc=ide-scsi
initrd /initrd-2.4.20-8.img
```

Of course you don't want to copy this verbatim, since your system will be different. This is only an example.

Starting the System

System 5 init

The most visible parts of the "Init" facility are:

/sbin/init and /etc/inittab program and it's configfuration data file

/etc/init.d scripts for starting and stoping processes

/etc/rc.d/ scripts and subdirectories rc1.d, rc2.d, ..., rc6.d

/etc/sysconfig/ additional "sources" and subdirectory network-scripts

On a RedHat 9.0 system, 'ls -R /etc/sysconfig/ /etc/init.d/ /etc/rc.d/ | wc -l' reports 671 entries!

Inittab

During boot the kernel locates /sbin/init. Init is the parent of all processes. Its primary role is to create processes from a script stored in the file /etc/inittab . The man page for inittab describes syntax of inittab lines ("id:runlevels:action:process")

runlevels: (0-6,A-C,or empty)

0 & 6 are used for shutting the machine down

1 is reserved for single-user mode

2-5 are available to define custom operating configurations

A-C are used for "ondemand" actions (*not* a runlevel change)

action values of particular interest (there are others, see man inittab)

initdefault	defines the default runlevel (eg "id:3:initdefault:")
sysinit	always processed, always before anything else (eg, "si::sysinit:/etc/rc/d/rc.sysinit")
once	used mainly for "any-level" processes (eg, "ud::once:/sbin/update") [starts bdflush]

wait	used for the primary runlevel-specifc action script (eg "i3:3:wait:/etc/rc.d/rc3")
respawn	– used to provide login gettys for virtual consoles, serial connections, xdm (eg, "6:2345:respawn:/sbin/mingetty tty6")
ctrlaltdel	process to be executed when the CTRL-ALT-DEL key combination (three finger salute) is pressed. Typically some shutdown procedure.

This line in /etc/inittab instructs /sbin/init what to do next:

si::sysinit:/etc/rc.d/rc.sysinit

rc.sysinit

1. reads ("sources") /dev/.devfsd checks to see if we're running devfs and starts the devfsd daemon
2. check to see if it exists then reads /etc/sysconfig/network for instructions on setting the hostname and starting networking
3. reads /etc/init.d/functions for instructions for setting up locale and for function definitions
4. checks to see if this will be a graphical boot and prepares for it
5. checks /etc/inittab and starts mgetty on configured virtual consoles
6. set console fonts
7. prints a banner to the console
8. mounts the /proc filesystem
9. unmounts the startup ramdisk
10. configures kernel paramters
11. sets the system clock
12. read he configuration and maps the keyboard for the terminals and console
13. sets the hostname
14. initializes USB devices and HID devices like USB keyboard and mouse and required USB storage
15. runs fsck on the file systems
16. checks for filesystem quotas and updates quotas on the root filesystem
17. checks for ISP plug-n-play devices and configures them

18. remounts the root filesystem in read-write
19. initializes LVM (Logical Volume Management)
20. activates swaps
21. writes a new mtab
22. starts syslog
23. runs depmd
24. load sound modules
25. load kernel modules
26. start any configured raid devices
27. initialize LVM again, in case any are on the raid just started
28. mount all other filesystems
29. check quotas on newly mounted filesystems
30. reread the network configuration and reset the hostname if necessary
31. clean up /var
32. reset the console permissions
33. clean up utmp
34. delete stale locks
35. turn on swap
36. read /etc/rc.serial and configure serial ports
37. check for scsi tape drives and load the module for it
38. load usb storage
39. check for and load firewire
40. load ide-scsi for things like ide CDR burners and DVD-R
41. run hdparm for ide drives
42. generate a header that defines the boot kernel
43. adjust symlinks for System.map file
44. dumps syslog messages to a file /var/log/dmesg
45. creates a crash indicator flag to warn on crashes

/etc/init.d/functions

1. sets the umask
2. sets up a default search path
3. configures the screen width
4. reads (sources) /etc/sysconfig/init which may be specific to Red Hat
5. checkpid() function to get the pid (process id) of a program
6. daemon() functions to start a program
7. killproc() function to stop a program
8. pidfileofproc() function to find the pid of a program
9. status() function to check whether a program is running or not
10. echo_success() function for reporting the successful start of a program during bootup
11. echo_failure() function for reporting the failure of the starting of a program
12. echo_passed() function to report the passing of a system test
13. echo_warning() function to report a warning
14. success() function logs that something succeeded, calls echo_success
15. failure() function logs the failure of some process, calls echo_failure
16. passed() function logs that something passed, possibly with errors calls echo_passed()
17. warning() function logs a warning, calls echo_warning()
18. action() run some action and log its output
19. confirm() function used during interactive startup to verify that you want service to start

Other scripts

Other scripts that get called from /etc/rc.d if they exist:

/etc/rc.d/rc.serial (may be needed to setup non-standard serial ports)

/etc/rc.d/rc.modules (comments say: ”for backward compatibility with VARs”) - if you ever find yourself tempted to edit this file, consider putting your additions in another script /etc/rc.local.

rc

/etc/rc.d/rc 3 - /etc/rc.d/rc is the script and "3" is a command-line argument telling it to process the runlevel-3 subdirectory (/etc/rc.d/rc3.d). It's responsible for starting and stopping services when the runlevel changes. Runs all "kill-scripts", and then all "start-scripts". Before running anything, it reads ("sources") /etc/rc.d/init.d/functions.

rcN.d (N=0,1,..,6) – the elegance revealed

All files in any rcN.d subdirectory are symbolic links, and with one exception, they all link to scripts in /etc/rc.d/init.d – for example S40crond links to /etc/rc.d/init.d/crond. The exception is the start script S99local (in runlevels 2-5 only), which is a link to /etc/rc.d/rc.local.

The default rc.local script, BTW doesn't do very much of great complexity; it only creates the file /var/lock/subsys/local on startup. It's only other remaining purpose is as a location for custom initializations for which the system administrator doesn't wish to create a complete startup script in /etc/init.d/ and that is desired in all (normal) runlevels.

update

from "ud::once:/sbin/update" - after the runlevel-specific configuration step above, the "any-runlevel" line from inittab gets run.

gettys

/sbin/mingetty tty1 (and similar "respawn" action lines) - the getty programs are started after everything else.

That's it! After the terminal gettys are started, you can login on any virtual console, or accept connections if any gettys were created on serial ports.

More about /etc/rc.d/init.d scripts

Typically the scripts in /etc/init.d follow a standard format, and commonly either start or kill a daemon. But they *could* do anything. The nfsfs script, for example mounts or unmounts all nfs filesystems that are defined in /etc/fstab. These scripts are generally easy to read and figure out. Most of them examine their command line input and do the appropriate thing corresponding to arguments of "start", "stop", "restart", "status". Sometimes there is a "reload" argument which may be either a synonym for restart, or perhaps a more efficient re-initialization. So, now you know a configuration trick: if you change

some configuration data file associated with (say) sendmail, you can ensure the changes are honored by executing the command '/etc/rc.d/init.d/sendmail restart' – don't forget the argument, "restart". Be aware that script names often duplicate the name of a binary executable somewhere in your path, so DON'T try to execute (for example) 'sendmail restart', this will not work!.

Managing runlevels – tools and references

Different Linux distributions use runlevels differently. One commonality is that runlevels 0 and 6 are reserved for shutdown and runlevel 1 is single-user mode. Single-user mode is generally used for doing system repairs. On Red Hat systems runlevel 2 is mult-user without networking, runlevel 3 is multi-user with networking, runlevel 4 is for custom configuration and runlevel 5 is multi-user with graphical login.

There are a at least three tools for configuring and managing runlevels on Red-Hat Linux systems. The graphical interface tool, redhat-config-services can be reached from Menu -> System Settings -> Server Settings -> Services. The default it to edit the currently running Runlevel. By choosing the Edit Runlevel tab, you can edit runlevel 3, 4 and 5. You can run man redhat-config-services for a synopsis on this program.

There is an ncurses based tool, for configuring runlevels. It too, will default to the current runlevel. However, according to the man page, you can edit different runlevels by calling it with the –level argument. Calling man ntsysv will give you more information for running the runlevel editor.

There is also a general purpose command line tool for managing runlevels, /sbin/chkconfig for querying and updating the runlevel system. Reading about chkconfig reveals some of the details about the mysterious numbers (S40, etc) in the scripts. Checkout also the chkconfig data that is embedded in the comments at the beginning of /etc/rc.d/init.d/ scripts (the ones that are "managed by chkconfig"). - /sbin/init (or /sbin/telinit) for changing runlevel, re-reading the inittab file, or activating "ondemand" processes.

See man init for details.

Packages and related docs

SysVinit provides the basic executable components of the SysV init scheme - man pages provided are last, lastb. mesg, wall, initscript, inittab, halt, killall, pidof,, poweroff, reboot, runlevel, shutdown, sulogin, telinit. Initscripts provides the basic scripts, utility programs, and the default runlevel links - man pages are doexec, ipcalc, netreport, usernetctl, usleep - requires the chkconfig package chkconfig provides the basic chkconfig tool and the curses wrapper program,

ntsysv both mentioned above. - man pages are chkconfig, ntsysv, redhat-config-packages provides the gui "Runlevel Editor" program, and some help text. - a man page exists for redhat-config-packages init and inittab

Berkeley init

Shutting Down the System

If the Linux system is not properly shutdown the file system can be corrupted. How corrupted depends upon what programs were being run, and whether the hard drives were synced before shutdown. Many time accidental shutdowns can be recovered, but sometimes not. It is better to know the proper methods than to risk file corruption. The shutdown command is the best method to insure programs are closed, files are saved and hard drives are properly synced. To shutdown and reboot, perhaps to start another OS, (gasp)

shutdown -r %time% # %time% can be now or a time in minutes

Many Linux systems trap the <Ctrl><Alt><Del> key combination (aka "Three Finger Salute") and alias it to "shutdown -r now" To shutdown, before electrically turning off the system, you might # shutdown -h now # you can substitute a number and specify time in minutes

System Management

Managing User Accounts

Jim Sack jsack@dornfeld.com

The passwd File

Jim Sack jsack@dornfeld.com

The group File

Jim Sack jsack@dornfeld.com

Adding User Accounts

Jim Sack jsack@dornfeld.com

Deleting and Disabling Accounts

Jim Sack jsack@dornfeld.com

Modifying User Accounts

Jim Sack jsack@dornfeld.com

Managing File Systems

As a system administrator, you can create, modify, and delete filesystems. Maintaining a filesystem is often not needed in a good filesystem.

Filesystem Basics

The most popular filesystem (based on the # of installations worldwide) is the FAT/FAT32 (File Allocation Table) filesystem used by Microsoft Windows - it is where the files are being stored and retrieved by the operating system and the user.

If you are using Windows and you partitioned your harddrive into several, or you have more than one harddrive, you will see C:, D:, E: drives, and so on. These are treated as different drives, and at the same time, different filesystems. For instance, D: can be a CD-ROM that contains StarCraft (a game), your E: drive might be a Zip drive. A CD-ROM uses the ISO9660 filesystem and can be read from Windows. A Zip diskette can have any filesystem, and once you format it in Windows, it will have FAT filesystem. your C: drive is also filesystem using FAT technology.

In other words, a filesystem is how a physical media (CD-ROM, Zip, Superdisk, DVD, Harddrive) is organized to store files. The operating system must support the particular filesystem in order to access data. Windows does not support Ext2 filesystem (used by Linux) but supports CIFS (Common Internet FileSystem) that enables using UNIX file servers on a Windows network, or visa versa.

A good filesystem provides or supports:

- good performance (defragmentation)
- minimized loss of data during crash (journaling)
- easy recovery of data from damaged filesystem
- flexibility (2-byte code filename, long filename, special files)
- easy management (self-diagnostics, tools)
- large capacity (Gigabytes, Terabytes)
- large file security (user id, group id, access control, permission)
- capability of being distributed (network filesystem)
- low to zero fragmentation

fragmentation: This happens a lot with Windows/DOS' FAT filesystem it refers to the phenomenon when a file is physicall divided and scattered around the disk, while logically seen as a one single file. Programs such as defrag.exe or Norton's Speeddisk regroups the files to provide optimal performance to the operating system. Linux has one, but it's obsolete since Ext2fs does this internally.

Journaling File Systems

If there is a system crash before buffers have been written to disk, it will cause the system to behave in an inconsistent way after reboot. A file deleted in the cache, may still remain on the hard disk. The fsck recover tool for ext2fs has to scan the entire disk partition in order to get the file system into a consistent state, after a crash. On a very large file system of serveral hundred gigabytes , getting the system back to a consistent state can take a very long time, delaying the availability of the server.

Journaling filesystems use a principal called Atomicity to maintain a consistency. What this means is that all operations belonging to a transaction look as if they are atomic operations that can't be partially performed. After every transaction, there must be a commit operation, making sure the buffers be written to disk. All operations are completed wihout errors, or canceled. All metadata is logged in a journaling file system. Metadata are the control structures inside a file system: i-nodes, free block allocation maps, i-nodes maps, etc. After a system crash, the log can be checked back to the first commit statement, writing the arguments content back to its position on the disk.

Filesystem Types

These are a few popular filesystems used in Linux. Ext2fs (Second Extended filesystem) - Linux calls this "ext2" This is de facto filesystem for Linux created by Remy Card, Wayne Davidson and others. Ext2fs provides good performance (especially when reading large files), long filenames, a set of management tools, UNIX-type security, low fragmentation, and read-ahead to name a few.

Ext2fs' maximum size is 8GB, and can have maxinum filesize of Xgb. Ext2fs is still under development (it doesn't mean that it is unstable but means that there are things that can be improved or implemented), and it's planned to include easy restoration of deleted file, Access Control List (for security), and automatic file compression. Some beta-quality variants of Ext2fs include Encrypted Ext2fs and journaling Ext2fs.

Ext3 The ext3 filesystem is a journaling extension to the standard ext2 filesystem on Linux. Journaling results in massively reduced time spent recovering a filesystem after a crash, and is therefore in high demand in environments where high availability is important, not only to improve recovery times on single machines but also to allow a crashed machine's filesystem to be recovered on another machine when we have a cluster of nodes with a shared disk. Maximum file system size 4 Terabytes, Maximum file size 2 Gigabytes.

DOS FAT - Linux calls these "msdos, umsdos, vfat" There are three variants: MSDOS, UMSDOS, and VFAT. Their core is based on the FAT filesystem. MSDOS is a traditional DOS file system with 14 character filename

contraint, and is the most simple variant of FAT filesystem. UMSDOS, which was not widely used and likely won't be, is a method of storing long filenames without making any change to the FAT structure itself. It's running on FAT also. VFAT is the latest filesystem used by Windows 95/98 to store long filenames. Linux supports FAT32(next FAT) implicitly. This filesystem does not have automatic defragmentation and wastes some space in management, but has high-recovery ratio (even recovers deleted files). Gotta loose FAT. :)

ISO9660 (cd-rom filesystem) - Linux calls this "iso9660" International Standard Organization (ISO) created this standard to use on CD-ROMs and every CD that stores data uses this. Microsoft implemented Joliet Extension so that a CD can hold music and video while still being compatible with regular CD player (i.e. BackStreet Boy's album -'EveryBody').

JFS provides fast file system restart in the event of a system crash. Using database journaling techniques, JFS can restore a file system to a consistent state in a matter of seconds or minutes, versus hours or days with non-journaled file systems. IBM's journaled file system technology, currently used in IBM enterprise servers, is designed for high-throughput server environments, key to running intranet and other high-performance e-business file servers. Maximum file system size is 4 to 32 petabytes, depending no the block size. Maximum files size also depends upon block size, from 512 Terabytes to 4 petabytes

NTFS (NT filesystem) - Linux calls this "ntfs" Used by Windows NT, Linux only can do read-only access to this filesystem at this moment - one of the reasons is that the security features Windows NT has that are not compatible with Unix/Linux. Windows NT can have two filesystems: FAT and NTFS. If NT is running on FAT, Linux can access the NT partition as a FAT/VFAT filesystem.

NFS (Network Filesystem) - Linux calls this "nfs" Network filesystem was developed by Sun Microsystems and still widely used to access remote filesystems. A great amount of research and details are involved in this, since a network is inherently unreliable, Transfering files take time, there are security, encryption, authentication problems and other challenges. Linux supports nfs, but it is still incomplete compared to some other operating systems. This is something most single user system administrators don't have to worry about.

proc the /proc directory contains files that are not part of any filesystem associated with your hard disks, CD-ROM, or any other physical storage device connected to your system (except, arguably, your RAM). Rather, these files are part of a virtual filesystem, enabled or disabled in the Linux kernel when it is compiled.

Reiserfs is a filesystem using a plug-in based object oriented variant on classical balanced tree algorithms. ReiserFS has fast journaling, which means that you don't spend your life waiting for fsck. ReiserFS does meta-data journaling, enabling fast crash recovery without the expense of full data journaling. maximum file system size 16 Terabytes. Maximum files size 2^10 petabytes.

Smbfs is a filesystem which understands the SMB protocol. This is the protocol Windows for Workgroups, Windows NT or LAN Manager use to talk to each other. It was inspired by samba, the program by Andrew Tridgell that turns any unix site into a file server for DOS or Windows clients.

XFS SGI's XFS filesystem for Linux is a high-performance journaling filesystem that provides rapid recovery from system crashes and the ability to support extremely large disk farms. XFS is licensed under the GPL Maximum files system size 18K petabytes, maximum file size 9K petabytes

Mounting Filesystems

Terms

Mount

"Mounting" a filesystem means making a filesystem accessible to the operating system. "Mounting point' is where the filesystem is being mounted such as '/'. This has tree structure.

Syntax

mount <options> source directory

The most useful option is "-t" which specifies the type of filesystem

you are mounting. Here are examples:

- mount -t msdos /dev/hda1 /mnt/dos
- mount -t vfat /dev/hda2 /mnt/windows
- mount -t iso9660 /dev/hdb /mnt/cd
- mount -t ext2 /dev/hdc1 /home
- mount -t nfs kernel-panic.com:/export /kernel-panic
- mount -t ext2 image img
- mount -t iso9660 cdimage linuxdist
- mount -t ntfs /dev/hdb1 /mnt/nt

Creating Filesystems

Make a filesystem

Let's suppose for a minute that I just added a new hard disk into my system. For our example, we'll assume that it's and 10Gb slave drive on the first ide channel. I want to create a place to store some of my important files. I decide that this will be under my home directory, so I create a directory that will become the mountpoint for a new filesystem called myfiles. So the full path to this directory is /home/myusername/myfiles. Before I can use this new drive, I need to create a filesystem.

fdisk I decide that I don't want to use the entire drive, so I'm going to partition it into 4 equal 2.5Gb primary partitions. They will become hdb1, hdb2, hdb3 and hdb4. I will use fdisk to partiton.

#fdisk /dev/hdb
The number of cylinders for this disk is set to 1653.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
(e.g., DOS FDISK, OS/2 FDISK)
Disk /dev/hdb: 255 heads, 63 sectors, 1653 cylinders
Units = cylinders of 16065 * 512 bytes
Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-621, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-621, default 621):
+2.5M
Command (m for help): n
Command action
e extended

p primary partition (1-4)

p Partition number (1-4): 2

First cylinder (413-1653, default 413):

Using default value of 413

Last cylinder or +size or +sizeM or +sizeK (413-1653, default 1653): +2.5Gb

Command (m for help): w

The partition table has been altered!

Calling ioctl() to re-read partition table.

#

I continue on in the same manner, creating the other two primary partitions, hdb3 and hdb4. When I have finished the partition table looks like this.

Disk /dev/hdb: 255 heads, 63 sectors, 1653 cylinders

Units = cylinders of 16065 * 512 bytes

Device	Boot	Start	End	Blocks	Id	System
/dev/hdb1		1	413	2441406	83	Linux
/dev/hdb2		414	827	2441406	83	Linux
/dev/hdb3		828	1240	2441406	83	Linux
/dev/hdb4		1241	1653	2441406	83	Linux

I write out the drive partition with the 'w' command in Linux which brings me back to the prompt.

mke2fs For this example I'm going to create an ext3 filesystem. The command I use is mke2fs, which by default creates an ext2 filesystem. Since ext3 is essentially an ext2 filesystem, with journaling, this is appropriate. Remember I'm going to create this partition to mount to /home/myusername/myfiles. So I use mke2fs like this:

mke2fs -b 1024 -j -L /myfiles /dev/hdb1

mke2fs 1.32 (09-Nov-2002)

Filesystem label=/home/myusername/myfiles

OS type: Linux

Block size=1024 (log=0)

Fragment size=1024 (log=0)

307200 inodes, 2449881 blocks

122494 blocks (5.00%) reserved for the super user

First data block=1 300 block groups 8192 blocks per group,

8192 fragments per group

1024 inodes per group

Superblock backups stored on blocks:

8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409, 663553,

1024001, 1990657

Writing inode tables: done

Creating journal (8192 blocks): done

Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 22 mounts or 180 days, whichever comes first. Use tune2fs -c or -i to override.

#

So the only thing left to do, is create an entry in /etc/fstab for my mount

```
# vi /etc/fstab
LABEL=/                 /                       ext3    defaults        1 1
LABEL=/boot             /boot                   ext3    defaults        1 2
none                    /dev/pts                devpts  gid=5,mode=620  0 0
LABEL=/home             /home                   ext3    defaults        1 2
none                    /proc                   proc    defaults        0 0
none                    /dev/shm                tmpfs   defaults        0 0
LABEL=/usr              /usr                    ext3    defaults        1 2
LABEL=/var              /var                    ext3    defaults        1 2
/dev/hda3               swap                    swap    defaults        0 0
/dev/fd0                /mnt/floppy             auto    noauto,owner,kudzu 0 0
/dev/cdrom              /mnt/cdrom              udf,iso9660 noauto,owner,kudzu,ro 0
0
LABEL=/myfiles          /home/myusername/myfiles/       ext3    de-
faults  0 0
```

I can manually mount the filesystem, or wait until I reboot. I prefer to mount it now.

```
# mount /home/myusername/myfiles
```

If I did everything correctly the mount command will check /etc/fstab find the file type and LABEL and mount the file without a problem. I can check with the mount command.

```
# mount
/dev/hda1 on / type ext3 (rw)
proc on /proc type proc (rw)
/dev/hda2 on /usr type ext3 (rw)
/dev/hda4 on /var type ext3 (rw)
none on /dev/pts type devpts (rw,mode=0622)
/dev/hdb1 on /home/myusername/myfiles type ext3 (rw)
#
```

It worked!

Creating Swap Space

The steps for creating swap space are similar to those for creating any other file system. One difference is the filesystem type under fdisk. We'll use the second partition on the drive above for our example. #fdisk /dev/hdb

```
The number of cylinders for this disk is set to 1653.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
Disk /dev/hdb: 255 heads, 63 sectors, 1653 cylinders
Units = cylinders of 16065 * 512 bytes
/dev/hdb1       1      413       2441406     83   Linux
/dev/hdb2     414      827       2441406     83   Linux
/dev/hdb3     828     1240       2441406     83   Linux
/dev/hdb4    1241     1653       2441406     83   Linux
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 82
Changed system type of partition 2 to 82 (Linux swap)
```

```
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
# fdisk -l /dev/hdb
Disk /dev/hdb: 255 heads, 63 sectors, 1653 cylinders
Units = cylinders of 16065 * 512 bytes
/dev/hdb1       1      413       2441406     83   Linux
/dev/hdb2     414      827       2441406     82   Linux swap
/dev/hdb3     828     1240       2441406     83   Linux
/dev/hdb4    1241     1653       2441406     83   Linux
# mkswap /dev/hdb
mkswap /dev/rd/c0d1p2
Setting up swapspace version 1, size = 2499999 kB
#
```

Enabling Swap Space

So now we need to add the swap partition to /etc/fstab so it's automatically enabled the next time we boot. We add the following line:

```
dev/hdb2              swap            swap   defaults          0 0
```

And we can enable the swap partition without rebooting by doing the following:

```
# swapon /dev/hdb2
#
```

Disabling Swap Space

To disable swap you just use the swapoff command with the same syntax as swapon.

```
# swapoff /dev/hdb2
#
```

ISO : mkisofs

Checking and Repairing Filesystems

fsck

Red Hat Control-Panel

The Red Hat control-panel is a gui interface, running under X, that allows easy administration of a Red Hat Linux System. Most of the configuration parameters can be adjusted using the control-panel. control-panel has to be run by the super-user, root. If you are running as a regular user, as recommended, you can use the su command in a xterm, to become root, then start control-panel from that xterm. If you just call control-panel you may get an error like this.

> $ su -
> Password:
> # control-panel
> ** WARNING **: cannot open display:
> To activate the control-panel from an xterm, where you su to root,
> take the following stepst.
> $ su -
> Password:
> # export DISPLAY=0:0
> # control-panel

Runlevel Editor

The Runlevel Editor can be called from the control-panel, or it can
be called by it's program name, tksyv.
$ su -
Password:
export DISPLAY=0:0
tksysv
A word of warning from the Help screen of tksysv:
Be warned, it does not cache changes. When you change something, it's
changed. It does store a backup of your /etc/rc.d tree in /tmp/tksysv/rc.d,

so you can restore it with:
cp -arf /tmp/tksysv/rc.d /etc
It makes this copy every time you run it, so don't messup, restart,
and then try and recover.
There is also more information about how to use the Runlevel Editor
in the help screen, once you have the application running.
When you start the Runlevel Editor you will be greeted by a panel consisting
of nine selection boxes and four command buttons. The left selection
box contains all the available system daemons or programs. This is
where you choose the applications you wish to have started. On the
right are eight selection panels representing runlevels two through
five, one for starting programs and snother for stopping. The runlevels
are as follows
Runlevel 2 Multi-user with no network active
Runlevel 3 Multi-user with networking active
Runlevel 4 Custom runlevel
Runlevel 5 Multi-user with networking and X, X login and X gui started
with system initialization
Runlevel 1, single user mode, is not configurable from the Runlevel
Editor, and Runlevel 6 is shutdown.
To add an application to a runlevel pick the application in the left
panel, then click on add. A new dialog box will appear. You must choose
whether you are editing the start or stopping of the daemon, and at
what runlevel your choices will effect.
To remove an application from a runlevel, simply click on the application
name, at the runlevel you wish to effect and then choose the Remove
button to prevent it from starting or stopping at that runlevel.
3.3.2 Time and Date
The control-panel contains a gui application, Red Hat calls timetool,
for setting the system time, and changing from twelve to twenty-four
hour clock conifguration. You can also call the application directly,

without the control-panel by doing the following:
$ su -
Password:
export DISPLAY=0:0
timetool

Printer Configuration

Printer configuration on a Linux system can be quite challenging. Fortunately Red Hat has recognized this and has integrated a set of print filters, manageable through this interface, that make it much simpler to produce high quality output even from older inkjet and dot matrix printers.
$ su -
Password:
export DISPLAY=0:0
printtool
Or you can call it from the standard control-panel, by clicking on the printer icon in the box. Printtool allows you to control print queues and four types of printers.
LOCAL printers are attached to the printer port of the local machine
REMOTE printers are attached to antoher machine, which has an lpd daemon running.
SMB printers are attached to an SMB or Windows or a samba server.
NCP printers are attached to a NCP. Use this to print on a machine running Netware mars_nwe software\
After a printer has been chosen you must choose the printer queue name, the spool directory and file limit, or maximum print job accepted for this printer.
For LOCAL printers you must also choose the printer device and the Input Filter, based upon what brand an model printer you have.
For REMOTE printers, you must also have the remote hostname and print

queue name.
For SMB printers you much also have, the Print Server Name, IP address, Printer name, Printer User, and Printer password.
For NCP printers also required are Printer Server Name, Printer User, and Printer Password.
There are excellent on-line help and troubleshooting guides available from printtool.

Network Configuration

Beside getting to the Network configuration tool from the control-panel, you can also call it by itself in the following manner:
$ su -
Password:
export DISPLAY=0:0
netcfg
From this gui utility you can configure hostname, Domain and the locations of nameservers. This is all done within the first screen that apears when you start netcfg.
To enable the ability of you local machine to share data with other systems on your network, when you don't have named running, you must enter the names and addresses of other machines on your network. The hosts tab takes you to that section. Here you can add the ip addresses, hostnames and nick names for your other networked computers.
You can edit interface configurations from the Interfaces tab. To add or delete ethernet devices, set IP addresses click on the tab and then choose from Add, Edit, Clone, Alias, Remove, Activate and Deactivate.
Routing can also be configured in the Network Convigurator. When you click on the Routing tab, you will see places to turn on and off packet forwarding (masquerading), add default routes, and add other static routes you may need for your configuration.

Modem Configuration

Kernel Daemon Configuration

Search Help System

Package Management

Red Hat Package Manager

The Red Hat Package Manager allows users to package source code for new software into source and binary form so that binaries can be easily installed and tracked through the rpm database and source can be rebuilt easily. The database of packages and files that can be used for verifying packages and querying for information about files and/or packages.

RPM is very flexible and easy to use. This is a brief overview of rpm. For more information you can consult the man pages for rpm, or you can visit the rpm web site at http://www.rpm.org.

Installing and Upgrading Packages with RPM

The basic method for installing a binary or source package with rpm is:

root@localhost # rpm -i [install-options] <package_file>

If this is an upgrade of an already installed package, you can choose the upgrade method, to remove the old package, while installing the new like this:

root@localhost # rpm -U [install-options] <package_file>

You may also specify the <package_file> as an ftp style URL, which will cause the package to be downloaded and then installed.

This will install the named package, but will give no feedback to the person (root) installing the package. To get more information you can use the the following options.

INSTALL AND UPGRADE OPTIONS

–**force** Same as using both –replacepkgs, –replacefiles, and –oldpackage.

-h, –hash Print 50 hash marks as the package archive is unpacked. Use with -v for a nice display.

–**oldpackage** Allow an upgrade to replace a newer package with an older one.

–**percent** Print percentages as files are unpacked from the package archive. This is intended to make RPM easy to run from other tools.

–**replacefiles** Install the packages even if they replace files from other, already installed, packages.

–**replacepkgs** Install the packages even if some of them are already installed on this system.

–**allfiles** Installs or upgrades all the missingok files in the package, regardless if they exist.

–**nodeps** Don't do a dependency check before installing to upgrading a package.

–**noscripts** Don't execute the preinstall or postinstall scripts.

–**notriggers** Don't execute scripts which are triggered by the installation of this package.

–**excludedocs** Don't install any files which are marked as docu- mentation (which includes man pages and texinfo documents).

–**includedocs** Install documentation files. This is only needed if excludedocs: 1 is specified in an rpmrc file.

–**test** Do not install the package, simply check for and report potential conflicts.

–**prefix** <path> This sets the installation prefix to <path> for relocatable packages.

–**ignorearch** This allows installation or upgrading even if the architectures of the binary RPM and host don't match.

–**ignoreos** This allows installation or upgrading even if the operating systems of the binary RPM and host don't match.

Conflicting Files

RPM keeps track of conflicts with already installed packages and will alert you of those conflicts. Under normal circumstances, unless the installer overrides with the –force option RPM will refuse to install the package.

Unresolved Dependencies

RPM knows about other software that the package being installed depend upon. It checks the system database of installed packages, and unless the installer overrides it with the –nodeps option, issues a dependancy warning, and does not install the package. It is generally a bad idea to override this dependancy check, unless you installed the dependant package, as a compiled binary or know of some other inormation.

Removing Packages with RPM

The command line for removing RPMs is:

rpm -e <package_file>

There are options for removing packages.

–**allmatches** Remove all versions of the package which match <package_name>. Normally an error is issued if <package_name> matches multiple packages.

–**noscripts** Don't execute the preuninstall or postuninstall scripts.

–**notriggers** Don't execute scripts which are triggered by the removal of this package.

–**nodeps** Don't check dependencies before uninstalling the packages.

–**test** Don't really uninstall anything, just go through the motions.

-vv option.

–**nodeps** Don't check for broken dependencies before removing the package.

Querying with RPM

You can query RPMs for information about the packages and you can query the rpm database about files on the system. The general form of an rpm query is:

rpm -q {query-options]

Some of the more popular forms are;

To find out the contents of a package, or which files and where they were put on the system.

rpm -qpl <package_file>

To query a file to find out what package it came from.

rpm -qf <file_name>

There are many options for querying packages and the database.

-a Query all installed packages

–**whatrequires** <capability> Query all packages that requires <capability> for proper functioning.

–**whatprovides** <virtual> Query all packages that provide the <virtual> capability.

-f <file> Query package owning <file>.

–**requiredby** <package> Query all of the package which contain triggers scripts that are triggered by <package>.

-p <package_file> Query an (uninstalled) package <package_file>. The <package_file> may be specified as an ftp style URL, in which case the package header will be down- loaded and queried. See FTP

OPTIONS for information on RPM's built in ftp support.

Information selection options:

-i Display package information, including name, ver- sion, and description. This uses the –queryformat if one was specified.

-R List packages this one depends on (same as –requires).

–**provides** List capabilities this package provides.

–**changelog** Display change information for the package.

-l List files in package.

-s Display the states of files in the package (implies -l). The state of each file is either normal, not installed, or replaced.

-d List only documentation files (implies -l).

-c List only configuration files (implies -l).

–**scripts** List the package specific shell scripts that are used as part of the installation and uninstallation processes, if there are any.

–**triggers** Display the trigger scripts, if any, which are contained in the package.

–**dump** Dump file information as follows: path size mtime md5sum mode owner group isconfig isdoc rdev sym- link. This must be used with at least one of -l, -c, -d.

Verifying with RPM

Verifying a package compares information about the installed files in the package with information about the files taken from the original package and stored in the rpm database. Among other things, verifying compares the size, MD5 sum, permissions, type, owner and group of each file. Any discrepencies are displayed. The package specification options are the same as for package querying.

Verifying rpms is used as a method of determing changes that may have been made to a system after a security breach. The form this takes is:

```
root@localhost # rpm -Va
```

This will report all changes to files since the original installation of packages. It requires some knowledge of the system to interpret, since many of the configuration files are changed immediately after an install. Files that were not installed from the package, for example documentation files excluded on installation using the ”–excludedocs” option, will be silently ignored.

The format of the output is a string of 8 characters, a possible ”c” denoting a configuration file, and then the file name. Each of the 8 characters denotes the result of a comparison of one attribute of the file to the value of that attribute recorded in the RPM database. A single ”.” (period) means the test passed. The following characters denote failure of certain tests:

5 MD5 sum

S File size

L Symlink

T Mtime

D Device

U User

G Group

M Mode (includes permissions and file type)

Signature Checking RPMs

RPMs can be produced with PGP signatures. All RPMs built by Red Hat for example are signed with Red Hat's PGP signature. This helps to insure the integrity and origin of the package. To check the signature of an RPM use this command:

```
rpm –checksig <package_file>
```

Other RPM Options and Resources

You may want to learn how to build your own RPMs, sign them with your PGP signature, rebuild your RPM database, or install packages throught the FTP options to RPM. Since this document is intended for beginning Linux users, we will not go into those options here. It's good to know that these options are all avaialbe to you. Here are some other resources about RPM, if you would like to learn more.

RPM Web Site http://www.rpm.org

RPM FTP Site ftp://ftp.rpm.org/pub

Maximum RPM a book by Ed Bailey. It's available in hardback and is 442 pages long and includes a quick reference card. The book covers everything from general RPM usage to building your own RPMs to programming with rpmlib. There is a link on the RPM Web Site where you can download the compressed file of the book in PostScript or LaTeX .

Red Hat Web Site http://www.redhat.com

System Configuration

Webmin

redhat-config

Lan Barnes

Upgrading Software and the Kernel

Archive and Compression Utilities

These are utilities for archiving. Utilities like gzip can be used with archiving utilities like tar to compress the file, before archiving. This allows more data to be stored on the archive media.

gzip, gunzip, zcat

gzip, gunzip, zcat - compress or expand files Gzip reduces the size of the named files using Lempel-Ziv coding (LZ77). Whenever possible, each file is replaced by one with the extension .gz, while keeping the same own ership modes, access and modification times. If no files are specified, or if a file name is "-", the standard input is compressed to the standard output. Gzip will only attempt to compress regular files. In particular, it will ignore symbolic links. By default, gzip keeps the original file name and timestamp in the compressed file. These are used when decompressing the file with the -N option. This is useful when the compressed file name was truncated or when the time stamp was not preserved after a file transfer.

Compressed files can be restored to their original form using gzip -d or gunzip or zcat. If the original name saved in the compressed file is not suitable for its file system, a new name is constructed from the original one to make it legal.

gunzip takes a list of files on its command line and replaces each file whose name ends with .gz, -gz, .z, -z, _z or .Z and which begins with the correct magic number with an uncompressed file without the original extension. gunzip also recognizes the special extensions .tgz and .taz as shorthands for .tar.gz and .tar.Z respectively. When compressing, gzip uses the .tgz extension if necessary instead of truncating a file with a .tar extension.

OPTIONS

-a –ascii Ascii text mode: convert end-of-lines using local conventions. This option is supported only on some non-Unix systems. For MSDOS, CR LF is converted to LF when compressing, and LF is converted to CR LF when decompressing.

-c –stdout –to-stdout Write output on standard output; keep original files unchanged. If there are several input files, the output consists of a sequence of independently compressed members. To obtain better compression, concatenate all input files before compressing them.

-d –decompress –uncompress Decompress.

-f –force Force compression or decompression even if the file has multiple links or the corresponding file already exists, or if the compressed data is read from or written to a terminal. If the input data is not in a format recognized by gzip, and if the option –stdout is also given, copy the input data without change to the standard ouput: let zcat behave as cat. If -f is not given, and when not running in the background, gzip prompts to verify whether an existing file should be overwritten.

-h –help Display a help screen and quit.

-l –list For each compressed file, list the following fields:

- compressed size:
- size of the compressed file
- uncompressed size: size of the uncompressed file
- ratio: compression ratio (0.0% if unknown)
- uncompressed_name: name of the uncompressed file

The uncompressed size is given as -1 for files not in gzip format, such as compressed .Z files. To get the uncompressed size for such a file, you can use:

```
zcat file.Z | wc -c
```

In combination with the –verbose option, the following fields are also displayed:

- method: compression method
- crc: the 32-bit CRC of the uncompressed data
- date & time: time stamp for the uncompressed file

The compression methods currently supported are deflate, compress, lzh (SCO compress -H) and pack. The crc is given as ffffffff for a file not in gzip format.

With –name, the uncompressed name, date and time are those stored within the compress file if present.

With –verbose, the size totals and compression ratio for all files is also displayed, unless some sizes are unknown. With –quiet, the title and totals lines are not displayed.

-L –license Display the gzip license and quit.

-n –no-name When compressing, do not save the original file name and time stamp by default. (The original name is always saved if the name had to be truncated.) When decompressing, do not restore the original file name if present (remove only the gzip suffix from the compressed file name) and do not restore the original time stamp if present (copy it from the compressed file). This option is the default when decompressing.

-N –name When compressing, always save the original file name and time stamp; this is the default. When decompressing, restore the original file name and time stamp if present. This option is useful on systems which have a limit on file name length or when the time stamp has been lost after a file transfer.

-q –quiet Suppress all warnings.

-r –recursive Travel the directory structure recursively. If any of the file names specified on the command line are directories, gzip will descend into the directory and compress all the files it finds there (or decompress them in the case of gunzip).

-S .suf –suffix .suf Use suffix .suf instead of .gz. Any suffix can be given, but suffixes other than .z and .gz should be avoided to avoid confusion when files are trans- ferred to other systems. A null suffix forces gun- zip to try decompression on all given files regardless of suffix, as in: gunzip -S "" * (*.* for MSDOS) Previous versions of gzip used the .z suffix. This was changed to avoid a conflict with pack(1).

-t –test Test. Check the compressed file integrity.

-v –verbose Verbose. Display the name and percentage reduction for each file compressed or decompressed.

-V –version Version. Display the version number and compilation options then quit.

-# –fast –best Regulate the speed of compression using the specified digit #, where -1 or –fast indicates the fastest compression method (less compression) and -9 or –best indicates the slowest compression method (best

compression). The default compression level is -6 (that is, biased towards high compres- sion at expense of speed).

tar

tar tricks

Upgrading Software

Upgrading Libraries

Upgrading the Compiler

Upgrading Other Software

Building a New Kernel

Why Compile a Kernel?

There are any number of reasons for building a custom kernel.

- You have found some new package that requires feature only available in the newer kernel.
- You have a special device on your system, that is only supported by the newer kernel.
- You heard it was a cool thing to do, so you want to try.
- You want to tune options in your kernel, to improve the performance of your system.
- You heard that the newest kernel version is the fastest ever, and you want to find out for yourself.
- A bug or security vulnerability has been discovered in the kernel version you are currently running, and you want to fix it.

Obtaining Kernel Sources

There are several ways to obtain kernel sources. You can install the sources that come with your distribution. If you are running Red Hat, you can use the Red Hat Package Manager, to install the sources from a rpm file. You can download the "tar ball" from the kernel ftp site or one of it's mirrors. The choice is yours.

Loadable Kernel Modules

Kernel modules are pieces of the kernel code which are not included directly in the kernel. They are compiled seperately, and they can be inserted and removed

from the running kernel at almost any time. So, they extend the capabilities of the kernel, while keeping the core components small and compact. Many popular devices, like the PCMCIA drivers and the ftape drivers are built as loadable modules. Rarely used options are often compiled as loadable modules, conserving memory while still allowing those options to be run. Loadable modules also allow a generic kernel to be built, that can run across a wide array of hardware configurations. Red Hat makes extensive use of loadable modules in their distributions.

Configuring for the compile

Once you chosen your kernel source, next step is to configure for your compile. This is just a simple outline of the steps. It will not include a discussion of the various options to configuring the kernel, those documents already exist, on your system. See the Kernel-HOWTO on your system, for more detailed information. It's usually located in the HOWTO directory at /usr/doc/HOWTO. You must be in the top level kernel source directory /usr/src/linux and you must be logged in as root, to compile a kernel. Again you have some choices.

make config You can answer questions in sequence from the command line by using the command

root@localhost # make config

One by one, you will be asked a series of questions to determine the options for options to prepare the configuration file /usr/src/linux/.config . The drawback to make config is that if you make a wrong choice you will have to start over. You can't go back with make config

make menuconfig This is a curses based program, similar to the graphical programs used to set up Red Hat Linux. To begin simply:

root@localhost # make menuconfig

This is a pretty straightfoward configuration interface, that allows you to go forward and backward through the configuration questions, until you have everything just right for your system. You can even run make menuconfig in a virtual console.

make xconfig This is an X based program for setting up you configuration file, before compiling your kernel.
There is a small trick for running make xconfig, if like me, you don't

run X as root. Open an xterm and su - to become root. If you run are running the default bash shell export DISPLAY=0:0 to allow you to run X applications as root while, logged into X as a regular user.
Of course you will still have to cd /usr/src/linux so you are in the right directory, then make xconfig will start the X interface to configure the kernel.

Compiling the Kernel

Now you are ready to do the kernel compile. Here are the commands.

```
root@localhost # make dep
root@localhost # make clean
root@localhost # make bzImage
root@localhost # make modules
root@localhost # make modules_install
```

However, you can save a little time, and simplify by stringing the commands together. There are two ways to string commands together on the command line, with the semi-colon ";" or the double ampersands "&&". The semi-colons don't impose any conditions on the execution of the next command in the sequence. This could potentially waste time, and cause unnecessary complications. Potentially the kernel compile step could fail, but the modules build and install. The double ampersand says, only execute the next command, in the sequence, if the previous command succeeded. So in the example above, the make modules command would fail, if the make bzImage command wasn't successful.

Here's what the command line would look like, if you string the commands together.

```
root@localhost # make dep && make clean && make bzImage &&
make modules && make modules_install
```

LILO

The new kernel is built and is in /usr/src/linux/arch/i386/boot/bzImage, time to move it into the /boot directory.

```
root@localhost # mv arch/i386/boot/bzImage /boot
```

Since we are already in the /usr/src/linux directory, we only need to use the relative path, not the entire path to the image. It's never recommended to replace a current version kernel, before testing. To do this run it first as linux.new and make sure there are no errors. So now it needs to be added to the /etc/lilo.conf.

Edit the file to look similar to this.

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=250
image=/boot/bzImage
label=linux.new
root=/dev/hda1
read-only
image=/boot/vmlinuz-2.4.20
label=linux
root=/dev/hda1
initrd=/boot/initrd-2.4.20.img
read-only
```

Then run lilo to include the new kernel. If it is successful something similar to this should appear on the screen.

```
root@localhost # lilo
Added linux.new *
Added linux
```

Now reboot the system, and choose linux.new to test the new kernel.

Other Administrative Tasks

Making Backups

Simple Backups

Incremental Backups

Scheduling Jobs Using Cron

What is Cron?

Cron is a daemon that executes scheduled commands listed in a crontab file. It checks every minute to see if there are ant jobs scheduled to run at that time, if so, it executes those jobs.

The cron deamon is usually started at bootup time.

What is crontab?

Crontab is the program used to install, deinstall or list the tables used to drive the cron daemon. Each user can have their own crontab, and though these are files in /var, they are not intended to be edited directly.

What does a crontab file look like?

The users crontab file looks like this. (man 5 crontab).

```
* * * * * command
| | | | |
| | | | - > Day of week, 0 or 7 Sunday, 1 Monday
| | | - - > Month 0-12
```

```
| | - - - > Day of month 0-31
| - - - - > Hour of day 0-23
- - - - - > Minute of hour 0-59
```

The system cron tab looks like the above, with an additional entries.

```
* * * * * user command
```

The typical Redhat crontab file looks like this. The times may be different.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 4 root run-parts /etc/cron.weekly
12 4 1 * * root run-parts /etc/cron.monthly
```

The MAILTO environment variable is used to specify where the output for that particular job should be sent to, if not set, it gets mailed to the owner of the crontab file.

The program run-parts is explained in another section, but what it does is execute every file with the +x (execute) bit set on programs listed in the directory as the option.

So every file in the /etc/cron.hourly will be executed 1 minute after each hour.

Every file in the /etc/cron.daily will be executed at 4:02 AM. You may notice that there is a user logging in at this time called nobody if you happen to monitor your logs. This is one if the programs in the cron.daily directory doing its job.

The cron.weekly is set to run Thursday at 4:22 AM and the cron.monthly is set to run at on the first day of the month at 4:22 AM.

Modifying the system crontab

If there are any programs you would like run at these scheduled times from the administrator account, its easy to just copy the program or batch file to that directory, you can even make a link to the program if you like. Otherwise, see the next section about creating a crontab file for a regular users account. To have your time updated every night, edit a file called /etc/cron.daily/z-synctime with the following.

```
/usr/bin/rdate -s time.ucla.edu
```

Then use the chmod command to make it executable.

```
chmod +x /etc/cron.daily/z-sync-time.
```

I prefixed the filename with a z, because the files are executed alphabetically, so this program will be executed last.

Creating your personnel crontab file

To create or edit your own personnel crontab file, use the command crontab -e. you should be in an editor defined by your $EDITOR environment variable (if not set you may be in the vi editor by default). If this is your first crontab file, you should have a blank file. Lets add something to execute every hour, on the quarter hour. Add this line.

```
15 * * * * echo "Test of my crontab" | mail -s "My Crontab" you-
rusername
```

Now save the file and quit your editor. Use the command crontab -l to view your crontab file.

At a quarter past the hour you should now have a new email with the subject 'My Crontab'. To remove your crontab use the crontab -r command. Note, this will remove your entire crontab file. If you would like to run a command at multiple times, say every 30 minutes instead of just once per hour, use a comma to separate the time sequences. An example of once per 1/2 hour would look like this.

```
15,45 * * * * echo "on the quarter hour and 3 quarter hour" | mail
yourusername
```

To use crontab as a weekly reminder, you could use the next example which sends me an email 15 minutes before Southpark starts on Saturday at 10 PM. The email is sent at 9:45PM.

```
45 21 * * 6 echo "SouthPark" | mail -s "TV SouthPark" davek
```

For the curious, the personnel crontabs files are located in the /var/spool/cron directory (on Redhat). These should not be edited manually in any case as the crontab deamon is notified of the changes when the crontab program edits the file.

run-parts explained.

Where is it?

Here's how to find the script on your system,

```
# which run-parts
/usr/bin/run-parts
```

And here's what it looks like before it's all discected in the piece by Jim Sack below. This is a good example of a shell script. Though the topic of shell scripts is beyond the purposes of this book, it does help illuminate how crontab is set up on a Red Hat Linux system, and give you a quick view of a shell script. If you are running a Red Hat system, you can view the file on your own system.

```
# cat /usr/bin/run-parts
#!/bin/bash
# run-parts - concept taken from Debian
# keep going when something fails
set +e
if [ $# -lt 1 ]; then
echo "Usage: run-parts <dir>"
exit 1
fi
if [ ! -d $1 ]; then
echo "Not a directory: $1"
exit 1
fi
for i in $1/* ; do
[ -d $i ] && continue
if [ -x $i ]; then
$i
fi
done
exit 0
```

The disection

James (Jim) G. Sack wrote this explanation of the program. The explanations are all covered in man bash or the bash reference manual.

```
# keep going when something fails
set +e
```

Normally a shell script does not exit on a command error. The -e option forces exit on error executing any command not otherwise tested. So the +e option reverts to default just in case the -e was in effect when *this* shell was invoked. Hence the "keep going when something fails". Admittedly, the action of the "+" is easy to overlook in reading man bash.

```
if [ $# -lt 1 ]; then
echo "Usage: run-parts <dir>"
exit 1
fi
```

if the argument-count (of args passed to this shell-script) is less than 1, then give a usage message and exit without doing anything else. The "fi" is the ending-keyword for the "if" command syntax. The semi-colon usage may be understood by giving the full syntax (ignoring elif/else variants) if condition; then command-list; fi plus explaining that semi-colons may be replaced with newines.

```
if [ ! -d $1 ]; then
echo "Not a directory: $1"
exit 1
fi
```

if the first argument ($1) is not (!) a directory-name (-d), give an error message and exit.

```
for i in $1/* ; do
[ -d $i ] && continue
if [ -x $i ]; then
$i
fi
done
```

All the ”sanity checks” are done, now the real meat: For every file (call it i) in the specified (from arg 1) directory, if it's a subdirectory, do nothing (just continue on with the next filename), else (if it's executable), execute it. It may be of interest to note that the ”for i in $1/*” gives fully-qualified filenames.

```
exit 0
```

This is just good script-programming practice. Always ensure that some meaningful return-code is returned, whatever the exit-path. Since this is the ”norma” (or ”successful”) return, the convention of returning 0 is appropriate. When this convention is observed, you can calls this script from within other commands and be sure of being able to test for success or failure – for this particular script, success is defined as having no syntax errors in the command args.

The at command

You can run a one time command with the at command. The at command will run a command at a specified time in the future. As an example suppose April 4 at 11:00 PM you start downloading a large number of files, through your 14.4 Kb modem. Your wife needs the phone at 6AM. So you run the following command.

```
at 06:00 apr 04 at pppoff at <EOT>
```

(EOT = the control key then the d key or ^d)

To send yourself a reminder to not burn dinner in 15 minutes, use the next sequence.

```
at now + 15 minutes at echo ”Dinner is burning” | mail yourusername
at <EOT>
```

You can use the 'now + # time-unit' to run a command from now until that amount of time. Where time-unit could be minutes, hours, days, or weeks. You can also use tomorrow or the British time frame ”teatime”. :=) See man at for more information.

Managing System Logs

Managing Print Services

Checking Printer Hardware

Controlling Printer Services with lpc

Printer Services Maintenance

Printer System Troubleshooting

Red Hat printtool

What to Do in an Emergency

Repairing Filesystems

Accessing Damaged Files

Restoring Files from Backup

Editors, Text Tools and Printing

Editing with vi

vi (short for 'visual', and pronounced vee-eye) is a screen-oriented editing program used by many Unix users and, in particular, programmers. Regular users of vi find it a very convenient editing program, but it does take learning, and at first that will look unintuitive. Hopefully this guide is but one step towards hopping over that hurdle.

Basic editing

Help! How do I get out of this confounded program?

:q!

vi has possibly one of the most unintuitive interfaces, and for the user who's used to graphical editors, the first thing they want to do is to quit and return to safety. :-) I heard stories of users who quit vi by rebooting their system! Of course, that doesn't help get your work done. The rest of this primer will, I hope, give you a slightly bigger picture. The sections below feature commands you will want to type.

Okay, so what useful things can I do other than quit?

i

Most users want to, of course, be able to add stuff to the file they're editing (that being one of the main uses of an editor). You can't just type straight away—when you start vi, you're in what may be called 'command mode' where you enter commands. Type the letter i (for insert), in lowercase. This tells the

editor that you want to insert text into your file (more precisely the 'editing buffer' since changes are not written back to the file unless explicitly told to; see the section on dealing with files). Just start typing from here on.

How do I get back into command mode?

<Esc>

When you're in insert mode, hitting the 'escape' key will get you back into command mode. Think of it as an escape from insert mode.

Help! The arrow keys don't work!

No, on some systems the arrow keys don't work. If you're currently in insert mode, get back into command mode (see above), then hit the keys corresponding to the direction you wish to move:

k to go up,

h to go left

l to go right

j to go down

Remember to get back into insert mode before you start entering text again.

How do I add text to the end of a line?

a

You'll probably realise that the direction keys will not bring the cursor beyond the last character in the line. Typing the letter a (for append) will put you in insert mode just to the right of the cursor. If you don't want to have to move to the end of the line to add something to the end of the line, type A in capitals. This will get you to the end of the current line, no matter where in the line you were.

But moving a character at a time is too slow!

w

b

Typing w in command mode moves you to the start of the next word, and b does the reverse.

^

$

Typing ^ gets you to the beginning of the line, and $ to the end. If you are familiar with regular expressions (outside the scope of this document) you'll find them easy to remember.

<Ctrl>U

<Ctrl>D

Control-U and Control-D move about half a screen up or down, respectively.

:1

:$

:1 gets you to the first line in the buffer. (It should be obvious by now that you can replace 1 with any line number you care to choose.) :$ gets you to the last line in the buffer.

I

A

If you're inserting text at the beginning or end of lines, the command I is similar to ^i (insert at beginning of line), and A is similar to $a (append at end of line).

I can't delete text in insert mode!

x

X

In insert mode, you can delete text that you've typed in the current line, in the current insertion. That is, if you hit the escape key and re-enter insert mode, you can't delete what you've typed previously by hitting the backspace key. One way to delete text, a character at a time, is to get into command mode and type either x or X the lower case version deletes the character at the cursor (and behaves similarly to the 'delete' key in graphical editors) and the upper case one deletes the character just before the cursor (and behaves similarly to the 'backspace' key in graphical editors).

d<motion>

Typing the letter d, followed by a motion command (like h or l, or more usefully w) deletes text from the cursor to the target of the motion command. For example, dw deletes from the cursor to the start of the next word.

dd

:d

Both of these commands delete the whole line. To delete a whole bunch of text I just hold down the d key until I'm satisfied.

Joining/splitting

:j

The :j command joins the current line with the next. A single space is added to the end of the first line, unless the first line ended with a full-stop, in which case two spaces are added. You split a line the conventional way go into insert mode, and hit the return key.

Substituting text

Yeah, to replace text you can just remove and then insert, but there are faster ways to do things. s Replaces the character under the cursor. More useful than it looks, when you read two sections down. S Replaces the whole line. :s/<from>/<to>/ Replaces the first instance of <from> in the line with <to>. <from> and <to> can both be normal plain text (e.g., :s/1998/1999/ replaces the first 1998 in the line to 1999) but <from> is more precisely described as a regular expression (which again is outside the scope of the document). :s/<from>/<to>/g As above, but replaces all instances and not just the first.

I'm looking for a needle in a haystack – help!

/<pattern>

This will search from the current cursor position, through to the end of the buffer, for <pattern> which is a regular expression (again, in most cases you can just type in plain text). ?<pattern> Searches backwards from the cursor position to the beginning of the buffer for the pattern.

But some of these commands only affect one character!

Most of the commands above that do not have a colon can have a number typed before it for repeating the action. Examples:

8w advances the cursor 8 words forward.

4s replaces the 4 characters from the cursor onwards.

3i inserts whatever text you type 3 times.

Useful if you need to repeat something many times.

10dd deletes 10 lines from the current line down.

Remember—the commands with colons do not behave that way.

So how do I repeat colon commands?

(This section is slightly more technical—you may wish to skip it.)

The colon commands (or more precisely called 'ex commands', because they are used in the ex(1) editor) work with the notion of 'ranges', which come just after the colon. Ranges are best described by examples. The :1 and :$ commands mentioned earlier are not really commands, but line specifications, which stand for the first line and the last line respectively.

In addition, the dot . stands for the current line. You can offset this by a certain number of lines by using + or - followed by a number. For example, $-3 refers to three lines up from the end, and .+5 refers to the fifth line down from the current one.

A range is specified by two line specifications separated by a comma. For example, 2,5 means between lines 2 and 5, including lines 2 and 5.

The percent symbol % is a shortcut for 1,$.

And now, some real-life examples.

:%d zaps the whole buffer.

:.+1,$-4d deletes everything from the next line down to the end of the buffer, except the last four lines. If your email signature is, say, four lines long, this is a convenient way to zap your message without zapping your signature with it.

:%s/foo/bar/g replaces all instances of foo with bar.

:.,.+3j joins the current line with the three following lines.

How do I repeat what I just did?

.

The dot . command repeats the last command you did that actually changed the buffer. Movement commands do not count.

Oops! I've made some fatally awful (or awfully fatal) mistake!

u The standard way to undo the last action.

Note that many implementations of vi only have a single level of undo, so hitting u twice is the

same as doing nothing at all (it just undid your undo). The Berkeley vi implementation, nvi, allows multiple level undo by typing dots (it repeats your undo). Most cabbage vi implementations, though, repeat your last command (before your undo) if you type dots.

Playing with files

How do I save to a file?

:w

:w <filename> The :w command saves the current buffer (analogous to 'save' in a graphical editor). If you specify <filename>, the contents of the buffer is written to a different file. If you specify a filename, you normally won't be able to specify an already-existing file. If you must overwrite that file anyway, then add a ! after the w.

:w! foo.bar saves the buffer to foo.bar whether or not foo.bar already exists.

How do I edit a different file?

:e <filename> :e! The :e command either reloads the current file, or the file specified if <filename> is given. If your buffer has been modified and not yet saved, you won't be able to reload the buffer unless you add a ! after the e.

:e! foo.bar start editing foo.bar, discarding changes currently made to the buffer.

How do I embed another file?

:r <filename> Reads the contents of <filename> and embeds it into the current buffer.

:r !<command> Executes <command> and embeds into the buffer the output generated by the command.

:r !fortune -s puts a short fortune into your buffer—my favourite way of making email signatures.

Gettin' outta here.

:wq ZZ Both of these do pretty much the same thing, save changes made to the buffer and exit vi.

:q! Quits vi, discarding all changes made to the editing buffer.

The Emacs Editor

Dave Cook

The Pico Editor

What is Pico.

Pico is a basic console based text editor. Pico is installed with the mail program Pine.

*** It can be installed without Pine I do believe, look into it.

What Pico is not.

Pico is not a programmers editor. It doesn't appear to be designed as a programmers editor, it can still be used to edit program files when neccassary but for debuging purposes, it lacks some abilites from other editors, jumping by line number is not one of it's features..

Pico does a fine job of editing most of the text files used in the Unix/Linux envirionment like the /etc/hosts file for instance. Pico works and gets the job done.

Starting Pico.

Pico is started with the command pico or pico <file_name>, where file_name is the file to edit. To edit your .bashrc file you would type pico .bashrc at the command prompt. This poses a problem though, when you want to edit a file that has a line length greater 80 characters. Since Pico was designed to be used with Pine, it defaults to use word wrap at 80 characters. This doesn't work to well with some Linux configuration files. To disable word wrapping when editing a file, use the -w option. To edit your /etc/hosts file with word wrap disabled you would type pico -w /etc/hosts. As default, I always start pico with the -w option. You can alias the command to always start pico with the -w option. At a command prompt. alias mypico='pico -w $1'. Now when you type mypico <file_name> pico will now will have word wrap disabled. The alias command is covered in more detail in another section of this document.

More startup options.

As noted above, pico is not really designed for a programming environment. In the few cases where you might be compilling a program and would need to jump to a particular line number to find the error of a program the +N option would come in handy, where N would be the line number you want your cursor to start on. Say your writing a program and your compiler complains about a syntax error on line 42. Starting Pico on the command line like this "pico +42 myprogram.c" would start pico on with the cursor oon line 42 of your myprorgram.c file.

-v The -v option will allow you to use pico as a file viewer instead of an editor. The programs more and less doa fine job of viewing text files in there own right, you can use Pico with the -v option to do the same thing but with more of the ease of use of the Pico editor.

-z Using ctrl-z on some programs within Unix allows you to suspend that program and return to the commmand line. By default, suspending Pico is disabled, to enable the use of suspension within Pico, use the -z option. To return back to the file you are editing after using ctrl-z within Pico use the "fg" command. The command "fg" is a basic unix command and may be covered in another section of this document.

Other options.

There are a few other options that can be used when starting Pico, use "man pico" to view these other options.

e, -d Rub out

-k cut text from here to end of the line

-r# set column width for justifying.

Experiment and add to this document.

Help!

First things first. Start pico hold down the "CTRL" key then the "g" key. You will then be transported to the help screen. The rest of these sections will assume that you read the help screen and assumes that you know what ^G means. Use ^V (you dont need the shift key for the uppercase) to view the next page(s).

To exit the help screens use ^X.

Whats that stuff at the top of the screen?

Did you read the first page of the help screen? It explains the status line at the top of the display. ^G for help again

Whats that stuff at the bottom of the display?

The help section ^G. Says that the bottom 2 lines is the list the available editing commands. The third line from the bottom, which is blank most of the time is used for informational messages and for additional command input. Lets look at each of these commands listed on the bottom two lines.

The Exit command, getting out of Pico

^X quit pico. Depending on one of the three states of the file you are editing, Pico may ask you a few questions before you exit the editor.

File not modified If the far right hand section of the status line at the top of the document does not display "Modified", hitting ^X will exit Pico without any prompting. And you will be returned to the command prompt.

File modified and pre-exists If you are editing a prevuouly existing file, you started pico with the command "pico <file_name>", and you have modified the file. The far right hand section of the status line will display "Modified". Entering ^X will dispay the following text in the Command Input Line, third from the bottom.

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ? If you enter "n" the file will not be changed and you will exit to the command promt. If you enter "y", the Command Input Line will change to

File Name to write : <file_name> At this point you can hit the enter key to save the file as the file name you prevuoisly loaded or you can change it using the delte key and typng in a new file name. You can also save it to another directoy by giving the path name commands like "/tmp/mynewfile" or "../mynewfile" etc..

To exit either of these operations use the ^C command to abort and return to the Pico editor.

File modified and its a new file If you started Pico without giving a file name on the command prompt and you have edited the file, when hitting ^X you are again prompted whether you would like to save the file as noted above, upone hiting "y" you will be given the Command Input Line with a blank file name. Type in the file name you would like to save the file, if you hit enter with no file name you will be brought back to the Pico editor with "Cancled" in the Notify window. "Exit Cancled: will be displayed of ^C is hit at this point.

Saving the file, WriteOut ^O Will save the file.

Saving a pre-existing file

Saving a new file

Justifying text

^J justify your text. You're editing a file and you don't have word wrap on. You notice a typographical error or deleted some text. Pico has of course previously word wraped your text and you don't like the results.

^U will undo what you just did. You can go back and forth between ^J and ^U, more on ^U later.

Inserting a file.

^R read a file. Want to add an allready exiting file to the file your currently editing? Use ^R. Place the cursor where you want to add a file and hit ^R. At the comamnd prompt, type in the name of the file you want to insert. It says Home directory, but it really means the current directory where you started Pico. If you want to include a file not in the current directory, just use the path like "../myfile" or "/tmp/myfile" or "~/myfile". If the file is in the current directory just type "myfile".

Searching.

^W Where is

Moving about.

Previous Page.

Next Page.

Cursor Keys.

Cut, Copy and Paste.

Cutting Text.

Uncutting Text.

Marking Text.

Where am I?

Current position.

Spell Checker.

Jed

Text and Document Processing

Printing

The X Window System

X Concepts

Hardware Requirements

Installing XFree86

Configuring XFree86

Xconfigurator

XF86Setup

xf86config

Running XFree86

Running Into Trouble

Customizing Your X Environment

Window Managers

FVWM

FVWM95

AfterStep

WindowMaker

Enlightenment

Enlightenment has to be one of the best looking Window managers for the X windows system. What it lacks in power it makes up in its wownderful appear-

ance. Recently released DR 15 is fast, configurable, and worth downloading.

Most window managers are based on something else: FVWM95 (windows95), Window Maker (NeXT), AmiWM (AmigaOS). Enlightenment is totally different than any of those. It is not based on any previous OS or anything like that.

Mainly coded by two people, Carsten Haitzler (rasterman), and Geoff Harrison (mandrake). Both under the age of 30, or so they want us to think. It is themeable to the point of unrecognition. Some themes are made to mimic operating systems or even other window managers.

Homepage http://www.enlightenment.org/

Main Themes Page http://e.themes.org/

KDE

Books and Bibliography

Unix: The Complete Reference System V Release 3, Stephen Coffin, Osborne McGraw-Hill, ISBN 0-07-881299-2

Introducing the UNIX System, Henry McGilton and Rachel Morgan, McGraw-Hill Book Company, ISBN 0-07-045001-3

Running Linux Second Edition, Matt Welsh and Lar Kaufman, O'Reilly & Associates, Inc, ISBN 1-56592-151-8

Learning the bash Shell, Cameron Newham and Bill Rosenblatt, O'Reilly & Associates, Inc., ISBN 1-56592-147-X

Learning the vi Editor, Linda Lamb, O'Reilly & Associates, ISBN 0-937175-67-6

Linux in a Nutshell, Second Edition, Ellen Siever, O'Reilly & Associates, ISBN: 1-56592-585-8

Open Sources : Voices from the Open Source Revolution O'Reilly & Associates, ISBN: 1-56592-582-3